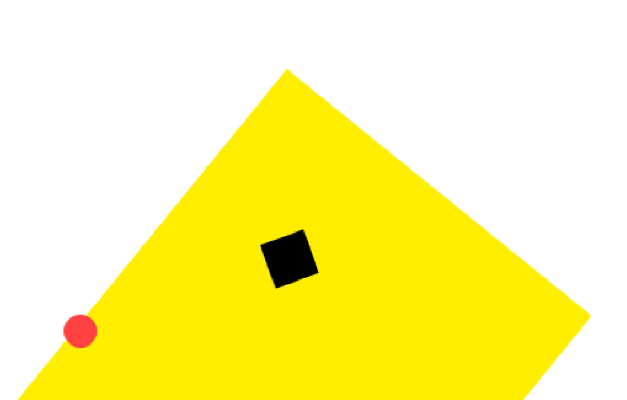
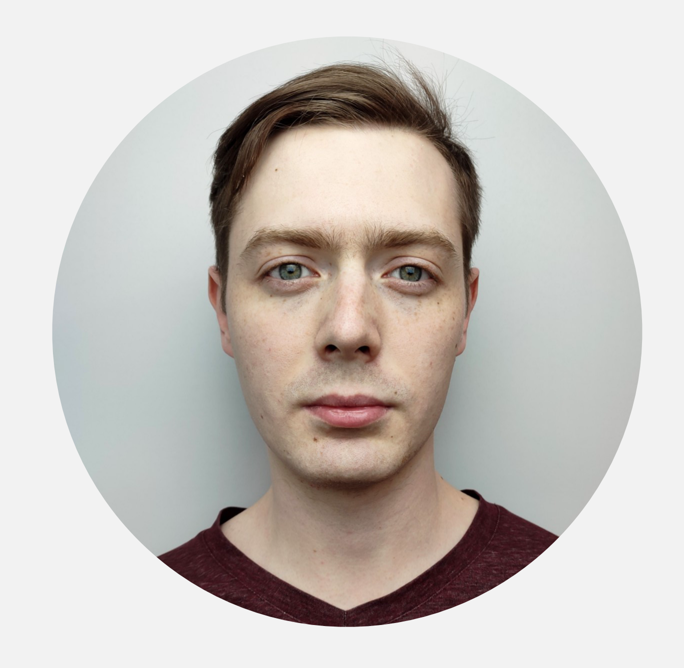
React Reconciler. 2D-игры на JSX

Сергей Константинов









Сергей Константинов

Raiffeisenbank - CFT



План

- Познакомить с Reconciler на примере ReactDOM
- Познакомиться с PIXI.js
- Подружить Reconciler с PIXI.js
- Посмотреть, как с помощью AST и кодогенерации можно собрать игровой движок





Предыстория



Предыстория

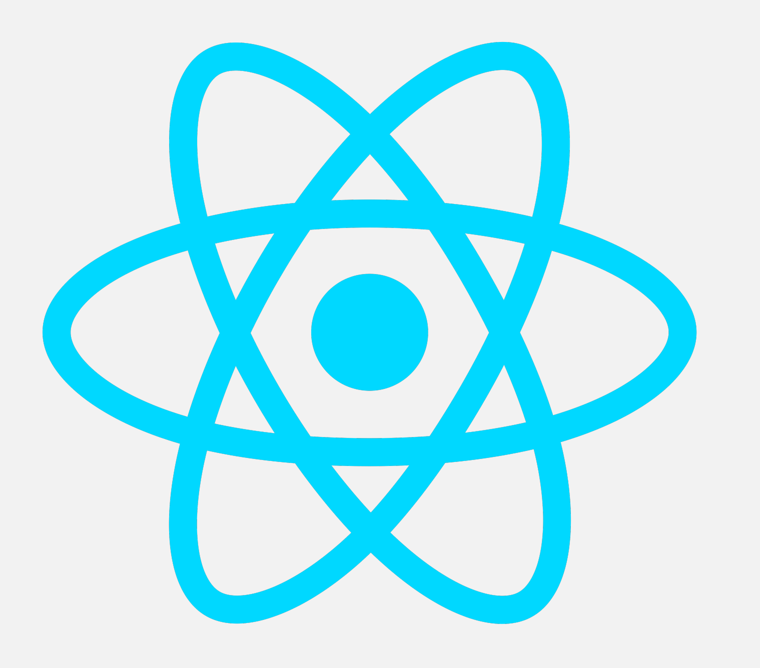
HTML5 Canvas змейка

Много бойлерплейта, сложно

Почему бы не писать игры

в декларативном стиле с JSX





React



React Reconciler

- Позволяет написать собственный рендерер
- Отделение алгоритма отрисовки от общих методов React'a
- Можно рендерить в DOM, Canvas, iOS, Android, терминал, figma, билборд, микроконтроллеры, можно даже писать музыку
- Инстанс PIXI.Application

jest-react	Fix duplicate words tests (#25333)	4 days ago
react-art	[Fizz/Float] Float for stylesheet resources (#25243)	11 hours ago
react-cache	Flow: upgrade to 0.140 (#25252)	18 days ago
react-client	[Flight] Implement error digests for Flight runtime and expose errorI...	8 days ago
react-debug-tools	[DevTools] Check if Proxy exists before creating DispatcherProxy (#25278	15 days ago
react-devtools-core	Fix: Documentation typos (#24471)	2 days ago
react-devtools-extensions	React DevTools 4.25.0 -> 4.26.0 (#25283)	15 days ago
react-devtools-inline	React DevTools 4.25.0 -> 4.26.0 (#25283)	15 days ago
react-devtools-shared	Fix duplicate words tests (#25333)	4 days ago
react-devtools-shell	Flow: well_formed_exports for devtools (#25266)	16 days ago
react-devtools-timeline	React DevTools 4.25.0 -> 4.26.0 (#25283)	15 days ago
react-devtools	Fix: Documentation typos (#24471)	2 days ago
react-dom-bindings	[Fizz/Float] Float for stylesheet resources (#25243)	11 hours ago
react-dom	[Fizz/Float] Float for stylesheet resources (#25243)	11 hours ago
react-fetch	Flow: remove explicit object syntax (#25223)	22 days ago
react-fs	Flow: remove explicit object syntax (#25223)	22 days ago
react-interactions	Flow: remove explicit object syntax (#25223)	22 days ago
react-is	straightford explicit types (#25253)	18 days ago
react-native-renderer	[Fizz/Float] Float for stylesheet resources (#25243)	11 hours ago
react-noop-renderer	[Fizz/Float] Float for stylesheet resources (#25243)	11 hours ago
react-pg	Flow: remove explicit object syntax (#25223)	22 days ago
react-reconciler	[Fizz/Float] Float for stylesheet resources (#25243)	11 hours ago
react-refresh	straightford explicit types (#25253)	18 days ago
react-server-dom-relay	[Fizz/Float] Float for stylesheet resources (#25243)	11 hours ago
react-server-dom-webpack	Move react-dom implementation files to react-dom-bindings (#25345)	2 days ago
react-server-native-relay	[Flight] Implement error digests for Flight runtime and expose errorI...	8 days ago
react-server	[Fizz/Float] Float for stylesheet resources (#25243)	11 hours ago



Reconciler

Renderer

hostConfig

ReactNative

CustomRenderer

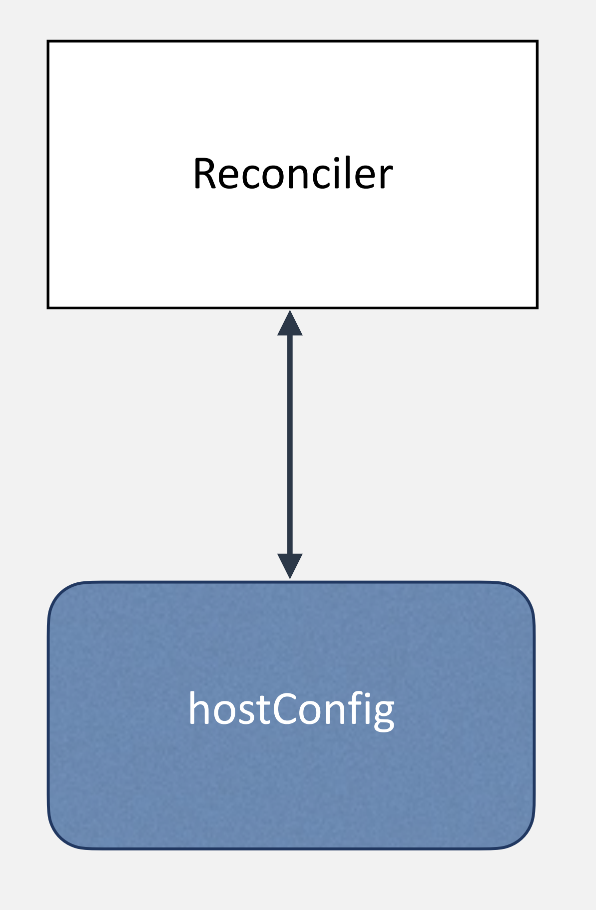
ReactDOM

Mobile

Terminal

Browser





Модуль react-reconciler с npm, реализующий алгоритм обработки дерева

Набор функций, специфичных для среды отрисовки. Например, создать, удалить или изменить узел



Познакомимся поближе

```
import React from 'react';
import ReactDOM from 'react-dom';

const root = document.getElementById('root');
ReactDOM.render(<h1>Hello, world!</h1>, root);
```



Познакомимся поближе

```
import React from 'react';
import ReactDOM from 'react-dom';

const root = document.getElementById('root');
ReactDOM.render(<h1>Hello, world!</h1>, root);
```

Компоненты, которыми оперирует рендерер – **host components**



Что такое JSX?

```
<h1 className='example'>Hello, world!</h1>

React.createElement(type, props, children)

React.createElement('h1', { className: 'example' }, 'Hello world')
```



ReactDOM.render(<App/>, root)

```
import Reconciler from 'react-reconciler';

const hostConfig = {}

export const render = (jsx, root) => {
    const reconciler = Reconciler(hostConfig);
    const container = reconciler.createContainer(root);
    reconciler.updateContainer(jsx, container, null, () => { });
}
```



ReactDOM.render(<App/>, root)

```
import Reconciler from 'react-reconciler';

const hostConfig = {}

export const render = (jsx, root) => {
    const reconciler = Reconciler(hostConfig);
    const container = reconciler.createContainer(root);
    reconciler.updateContainer(jsx, container, null, () => { });
}
```



ReactDOM.render(<App/>, root)

```
import Reconciler from 'react-reconciler';

const hostConfig = {}

export const render = (jsx, root) => {
    const reconciler = Reconciler(hostConfig);
    const container = reconciler.createContainer(root);
    reconciler.updateContainer(jsx, container, null, () => { });
}
```



ReactDOM.render(<App/>, root)

```
import Reconciler from 'react-reconciler';

const hostConfig = {}

export const render = (jsx, root) => {
    const reconciler = Reconciler(hostConfig);
    const container = reconciler.createContainer(root);
    reconciler.updateContainer(jsx, container, null, () => { });
}
```



ReactDOM.render(<App/>, root)

```
import Reconciler from 'react-reconciler';

const hostConfig = {}

export const render = (jsx, root) => {
    const reconciler = Reconciler(hostConfig);
    const container = reconciler.createContainer(root);
    reconciler.updateContainer(jsx, container, null, () => { });
}
```



ReactDOM.render(<App/>, root)

```
import Reconciler from 'react-reconciler';

const hostConfig = {}

export const render = (jsx, root) => {
    const reconciler = Reconciler(hostConfig);
    const container = reconciler.createContainer(root);
    reconciler.updateContainer(jsx, container, null, () => { });
}
```



ReactDOM.render(<App/>, root)

```
import Reconciler from 'react-reconciler';

const hostConfig = {}

export const render = (jsx, root) => {
    const reconciler = Reconciler(hostConfig);
    const container = reconciler.createContainer(root);
    reconciler.updateContainer(jsx, container, null, () => { });
}
```



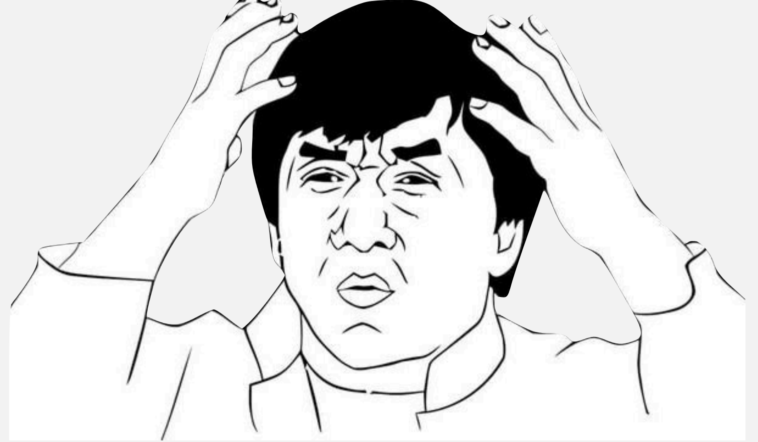
hostConfig = { … } – объект с набором методов, которые необходимо реализовать для того, чтобы reconciler мог взаимодействовать со средой, в которую рендерит

```
HostConfig.getPublicInstance
HostConfig.getRootHostContext
HostConfig.getChildHostContext
HostConfig.prepareForCommit
HostConfig.resetAfterCommit
HostConfig.createInstance
HostConfig.appendInitialChild
HostConfig.finalizeInitialChildren
HostConfig.prepareUpdate
HostConfig.shouldSetTextContent
HostConfig.shouldDeprioritizeSubtree
HostConfig.createTextInstance
HostConfig.scheduleDeferredCallback
HostConfig.cancelDeferredCallback
HostConfig.setTimeout
HostConfig.clearTimeout
HostConfig.noTimeout
HostConfig.now
HostConfig.isPrimaryRenderer
HostConfig.supportsMutation
HostConfig.supportsPersistence
HostConfig.supportsHydration
```

```
// -------------------
//     Mutation
//     (optional)
// -------------------
HostConfig.appendChild
HostConfig.appendChildToContainer
HostConfig.commitTextUpdate
HostConfig.commitMount
HostConfig.commitUpdate
HostConfig.insertBefore
HostConfig.insertInContainerBefore
HostConfig.removeChild
HostConfig.removeChildFromContainer
HostConfig.resetTextContent
HostConfig.hideInstance
HostConfig.hideTextInstance
HostConfig.unhideInstance
HostConfig.unhideTextInstance
```

```
// -------------------
//     Persistence
//     (optional)
// -------------------
HostConfig.cloneInstance
HostConfig.createContainerChildSet
HostConfig.appendChildToContainerChildSet
HostConfig.finalizeContainerChildren
HostConfig.replaceContainerChildren
HostConfig.cloneHiddenInstance
HostConfig.cloneUnhiddenInstance
HostConfig.createHiddenTextInstance
```

```
// -------------------
//     Hydration
//     (optional)
// -------------------
HostConfig.canHydrateInstance
HostConfig.canHydrateTextInstance
HostConfig.getNextHydratableSibling
HostConfig.getFirstHydratableChild
HostConfig.hydrateInstance
HostConfig.hydrateTextInstance
HostConfig.didNotMatchHydratedContainerTextInstance
HostConfig.didNotMatchHydratedTextInstance
HostConfig.didNotHydrateContainerInstance
HostConfig.didNotHydrateInstance
HostConfig.didNotFindHydratableContainerInstance
HostConfig.didNotFindHydratableContainerTextInstance
HostConfig.didNotFindHydratableInstance
HostConfig.didNotFindHydratableTextInstance
```





```
const hostConfig = {
  supportsMutation: true,
  now: Date.now,
  getRootHostContext: () => {},
  prepareForCommit: () => {},
  resetAfterCommit: () => {},
  getChildHostContext: () => {},
  shouldSetTextContent: () => {},
  createInstance: () => {},
  createTextInstance: () => {},
  appendInitialChild: () => {},
  finalizeInitialChildren: () => {},
  appendChildToContainer: () => {}
}
```

Reconciler

- UI agnostic
- Алгоритм обновления дерева элементов
- Оперирует методами hostConfig для работы с конкретной средой



Renderer

- UI dependent
- Оперирует хост-компонентами – div, h1, span
- Атомарно применяет изменения в конкретной среде



Режимы работы



Режимы работы

supportsMutation: true

Мутирует существующие ноды:

- appendChild, insertBefore, removeChild
- commitUpdate

Например: React DOM ...



Режимы работы

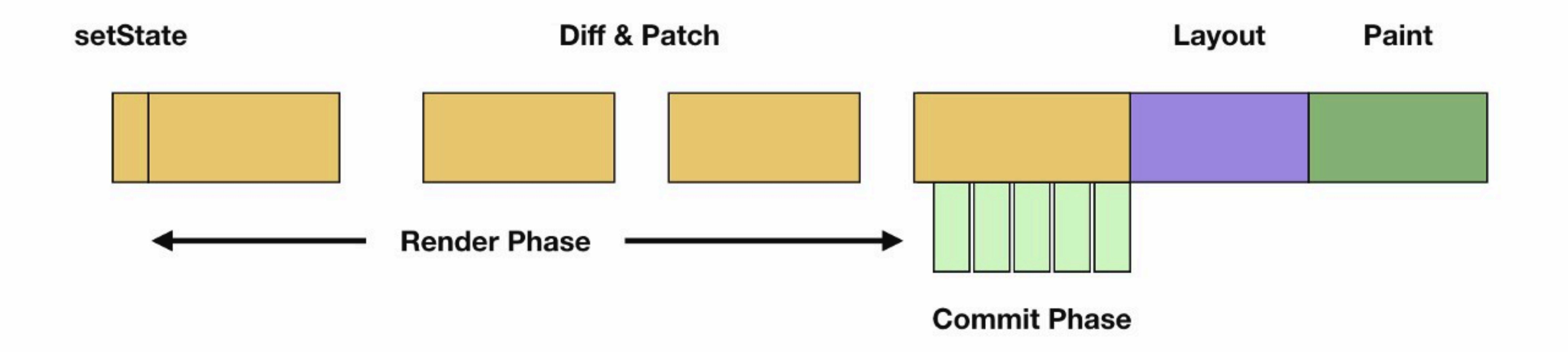
supportsPersistence: true

Клонирует и заменяет ноды:

- replaceContainerChildren, createContainerChildSet
- cloneInstance

Например: React Native, canvas, console, …



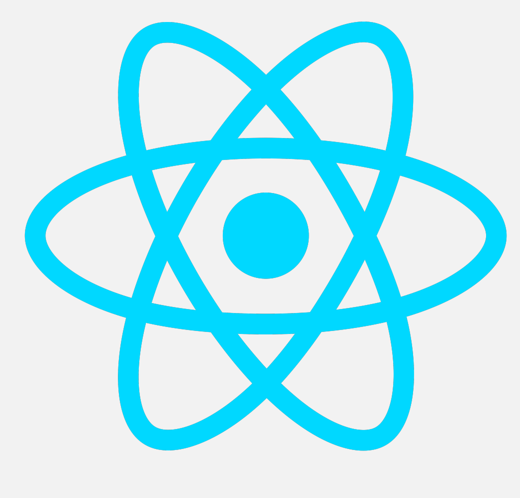

setState
Diff & Patch
Layout
Paint
Render Phase
Commit Phase

Первоначальная отрисовка



Первоначальная отрисовка

```
const App = () => (
  <div className='example'>
    <img className='image' src={logo} />
    <p>Hello!</p>
  </div>
);
```

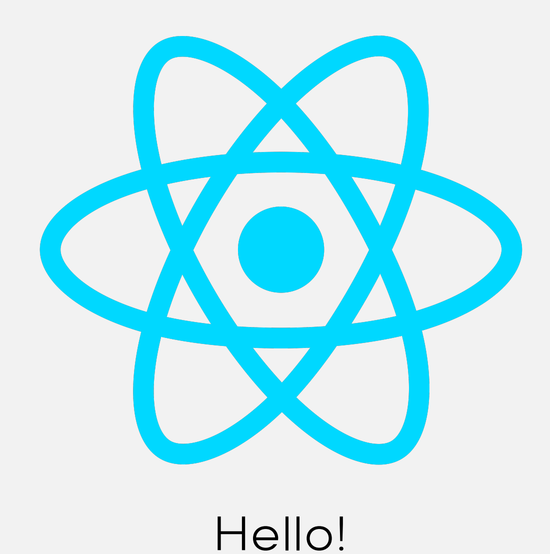


Hello!



Первоначальная отрисовка

```
const App = () => (
  <div className='example'>
    <img className='image' src={logo} />
    <p>Hello!</p>
  </div>
);
```

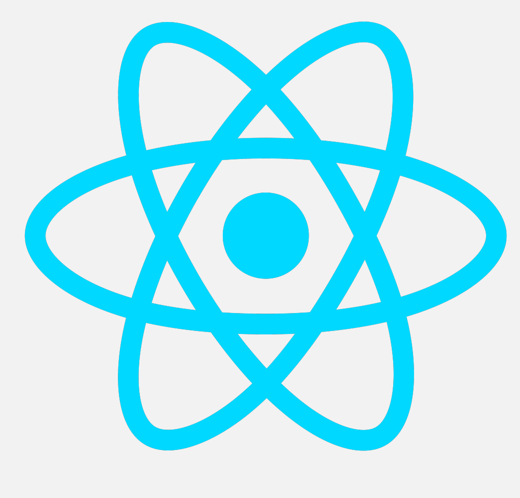


Hello!



Первоначальная отрисовка

```
const App = () => (
  <div className='example'>
    <img className='image' src={logo} />
    <p>Hello!</p>
  </div>
);
```

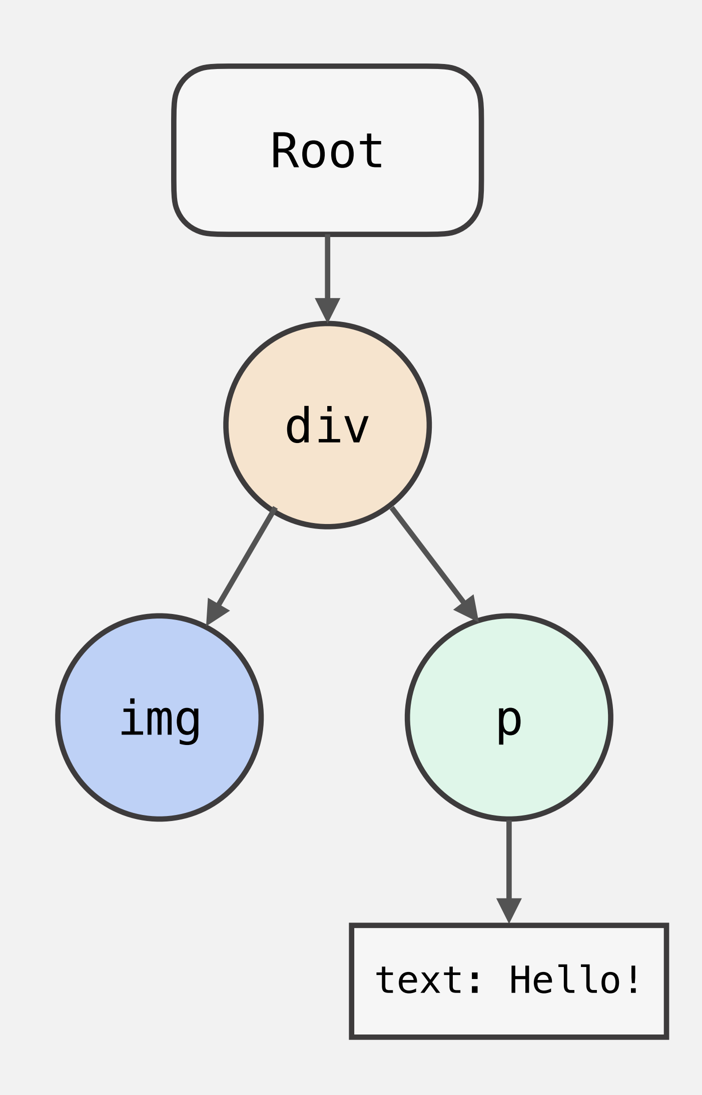


Hello!

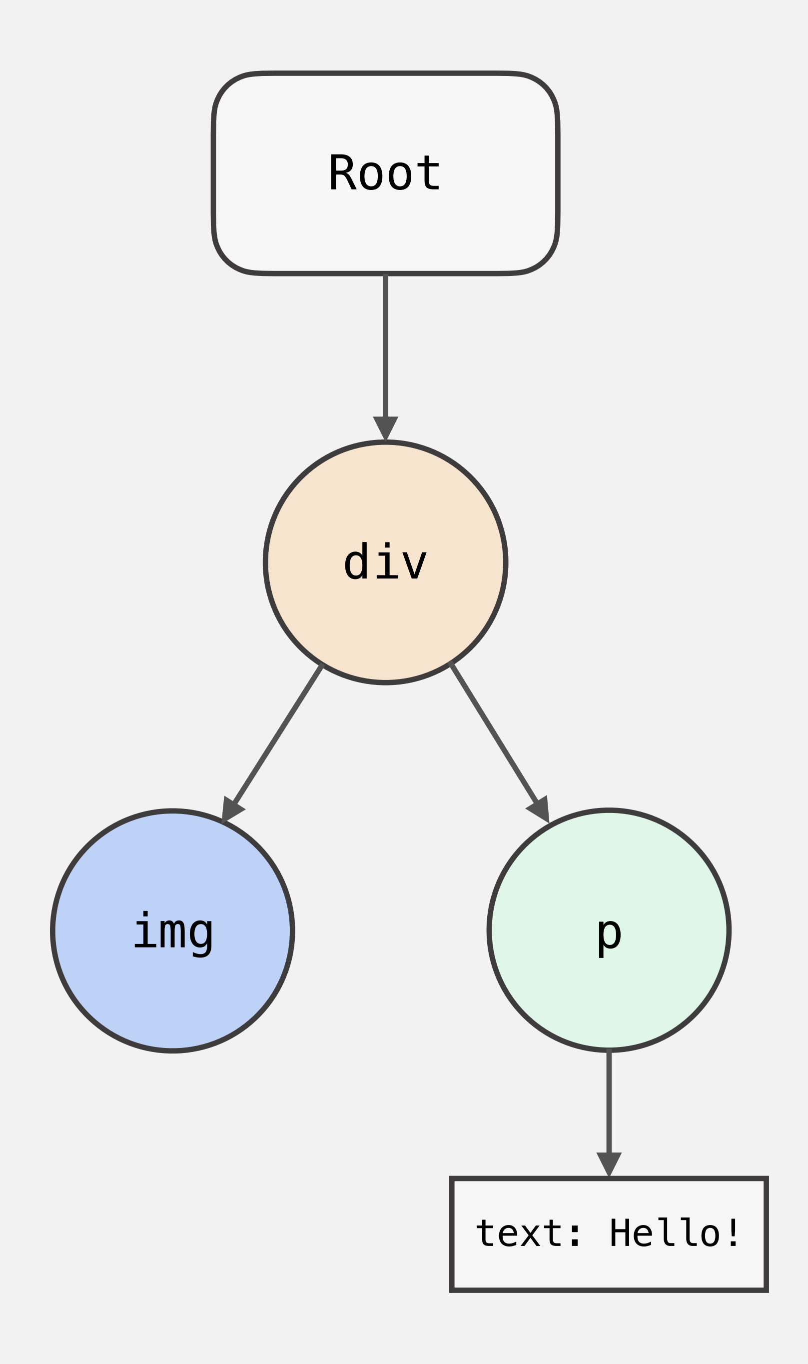


Первоначальная отрисовка

```
const App = () => (
  <div className='example'>
    <img className='image' src={logo} />
    <p>Hello!</p>
  </div>
);
```







Root
div
img
p
text: Hello!



FC
Frontend
Conf 2022

1. getRootHostContext

Root

div

img

p

text: Hello!



Root

div

img

p

text: Hello!

1. getRootHostContext

2. getChildHostContext



Root

div

img

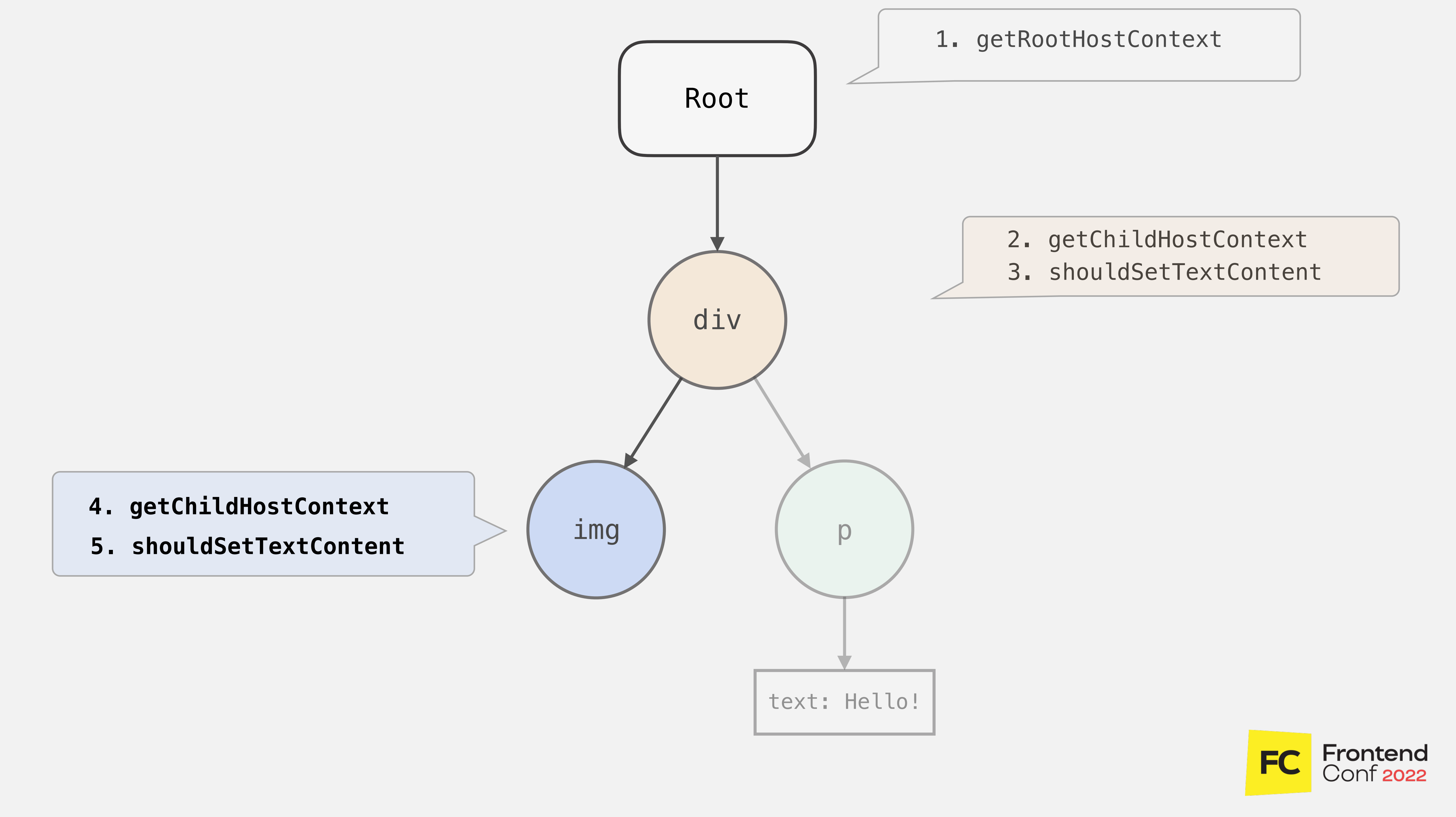
p

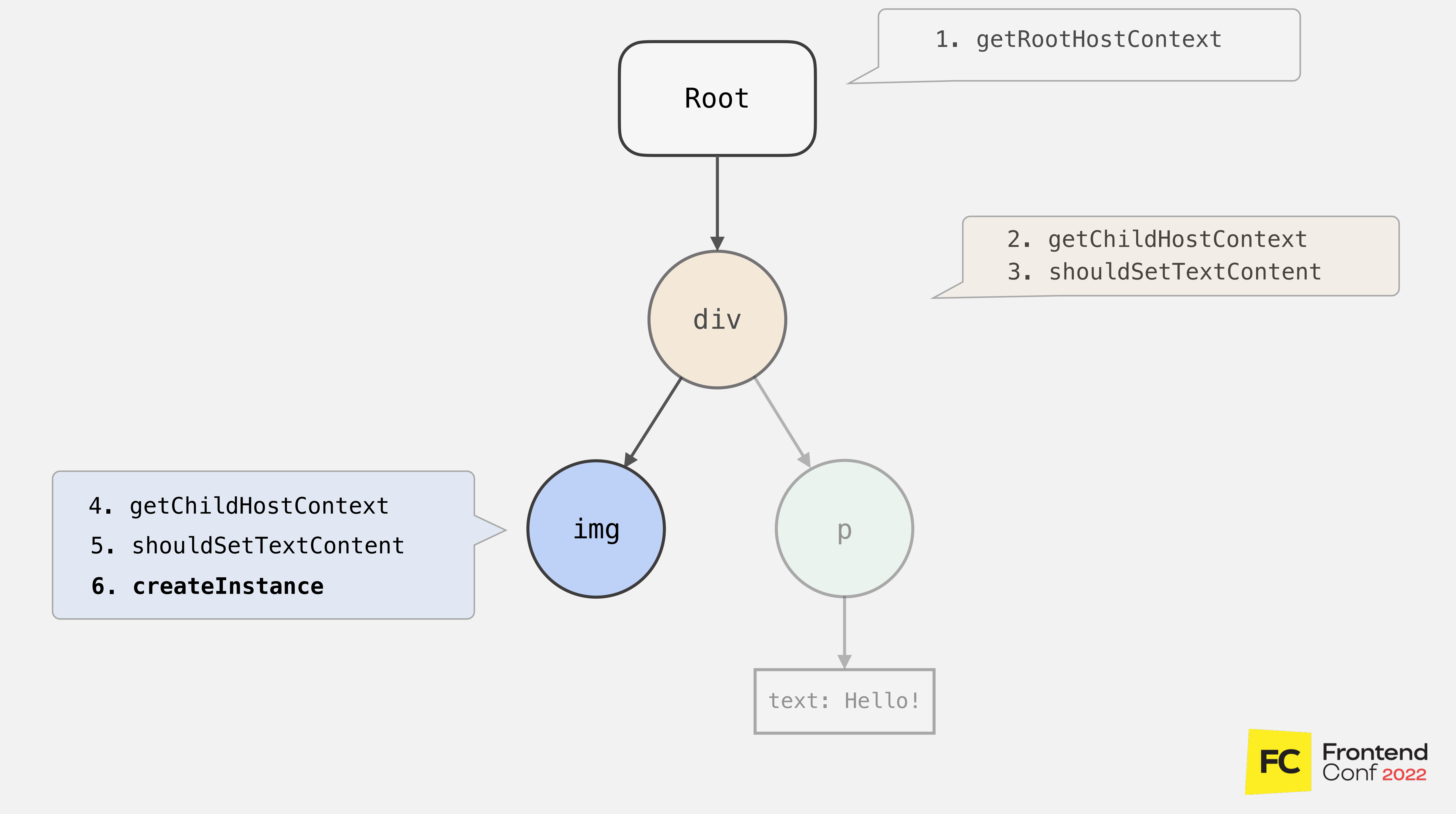
text: Hello!

1. getRootHostContext

2. getChildHostContext
3. shouldSetTextContent




1. getRootHostContext
Root
2. getChildHostContext
3. shouldSetTextContent
div
4. getChildHostContext
5. shouldSetTextContent
img
p
text: Hello!
FC
Frontend
Conf 2022

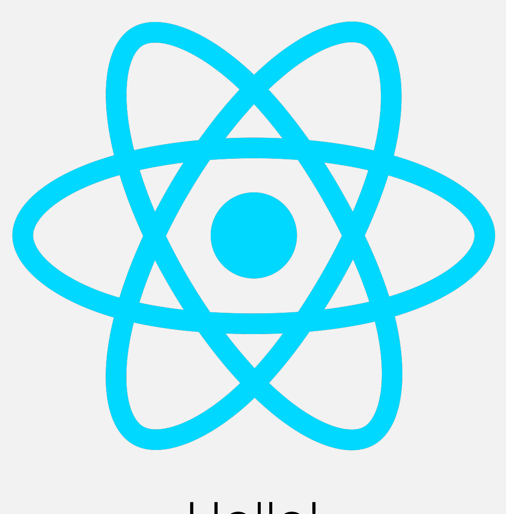

1. getRootHostContext
Root
2. getChildHostContext
3. shouldSetTextContent
div
4. getChildHostContext
5. shouldSetTextContent
6. createInstance
img
p
text: Hello!
FC
Frontend
Conf 2022

hostConfig

```
createInstance: (type, props) => {
    const instance = document.createElement(type);
    if (props.className) {
        instance.className = props.className;
    }

    if (instance.tagName === 'IMG' && props.src) {
        instance.src = props.src;
    }

    return instance;
}
```



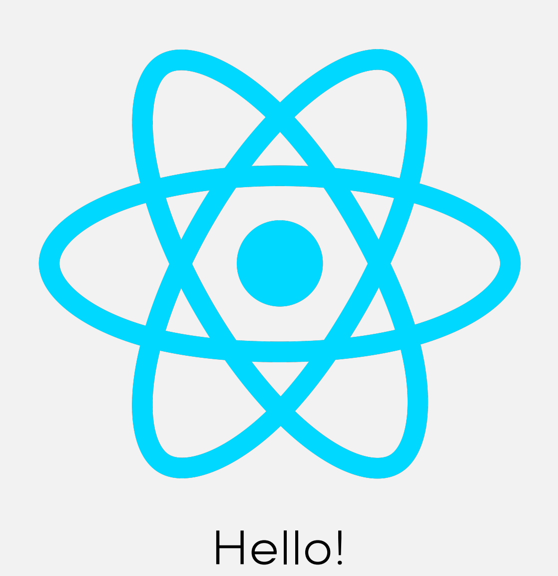


hostConfig

```
createInstance: (type, props) => {
    const instance = document.createElement(type);
    if (props.className) {
        instance.className = props.className;
    }

    if (instance.tagName === 'IMG' && props.src) {
        instance.src = props.src;
    }

    return instance;
}
```

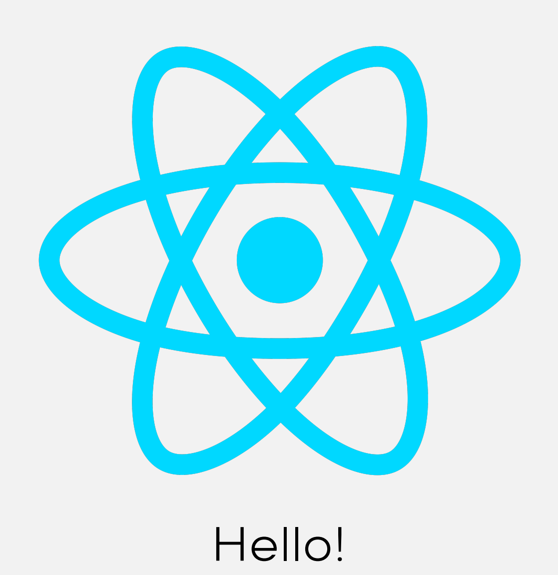


hostConfig

```
createInstance: (type, props) => {
    const instance = document.createElement(type);
    if (props.className) {
        instance.className = props.className;
    }

    if (instance.tagName === 'IMG' && props.src) {
        instance.src = props.src;
    }

    return instance;
}
```

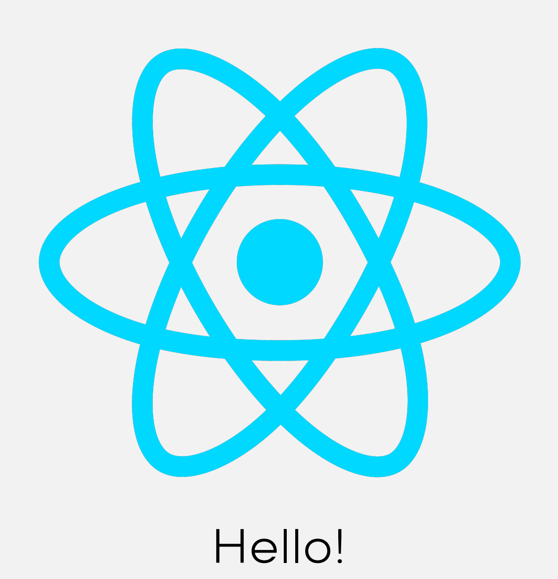


hostConfig

```
createInstance: (type, props) => {
    const instance = document.createElement(type);
    if (props.className) {
        instance.className = props.className;
    }

    if (instance.tagName === 'IMG' && props.src) {
        instance.src = props.src;
    }

    return instance;
}
```

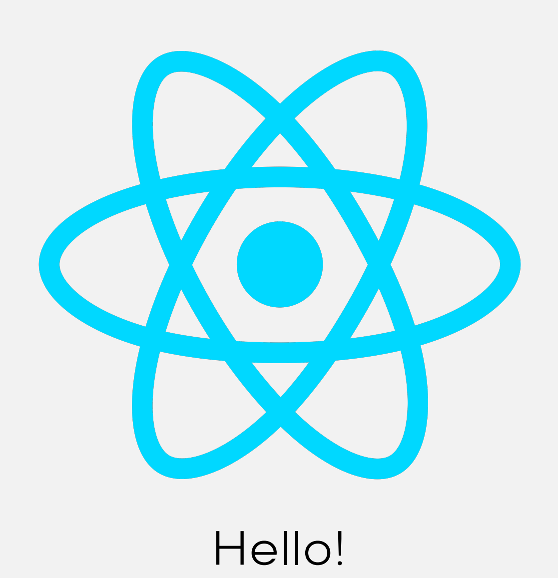


hostConfig

```
createInstance: (type, props) => {
    const instance = document.createElement(type);
    if (props.className) {
        instance.className = props.className;
    }

    if (instance.tagName === 'IMG' && props.src) {
        instance.src = props.src;
    }

    return instance;
}
```



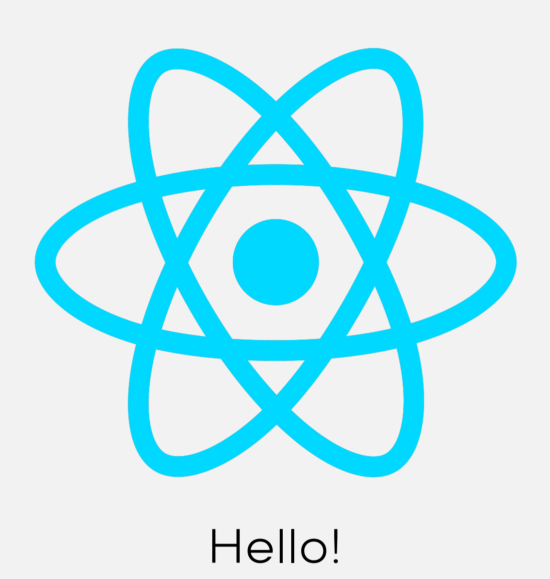


hostConfig

```
createInstance: (type, props) => {
    const instance = document.createElement(type);
    if (props.className) {
        instance.className = props.className;
    }

    if (instance.tagName === 'IMG' && props.src) {
        instance.src = props.src;
    }

    return instance;
}
```



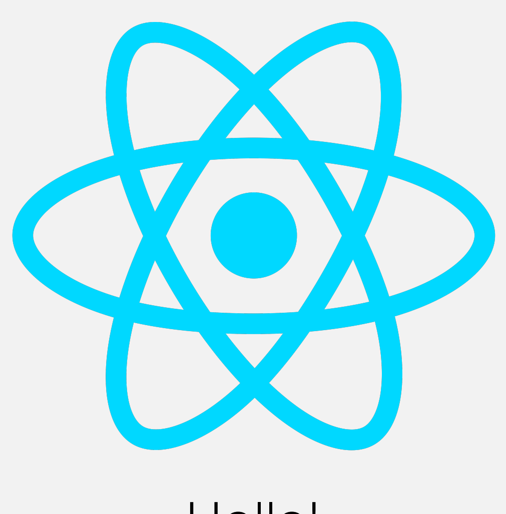
Hello!

hostConfig

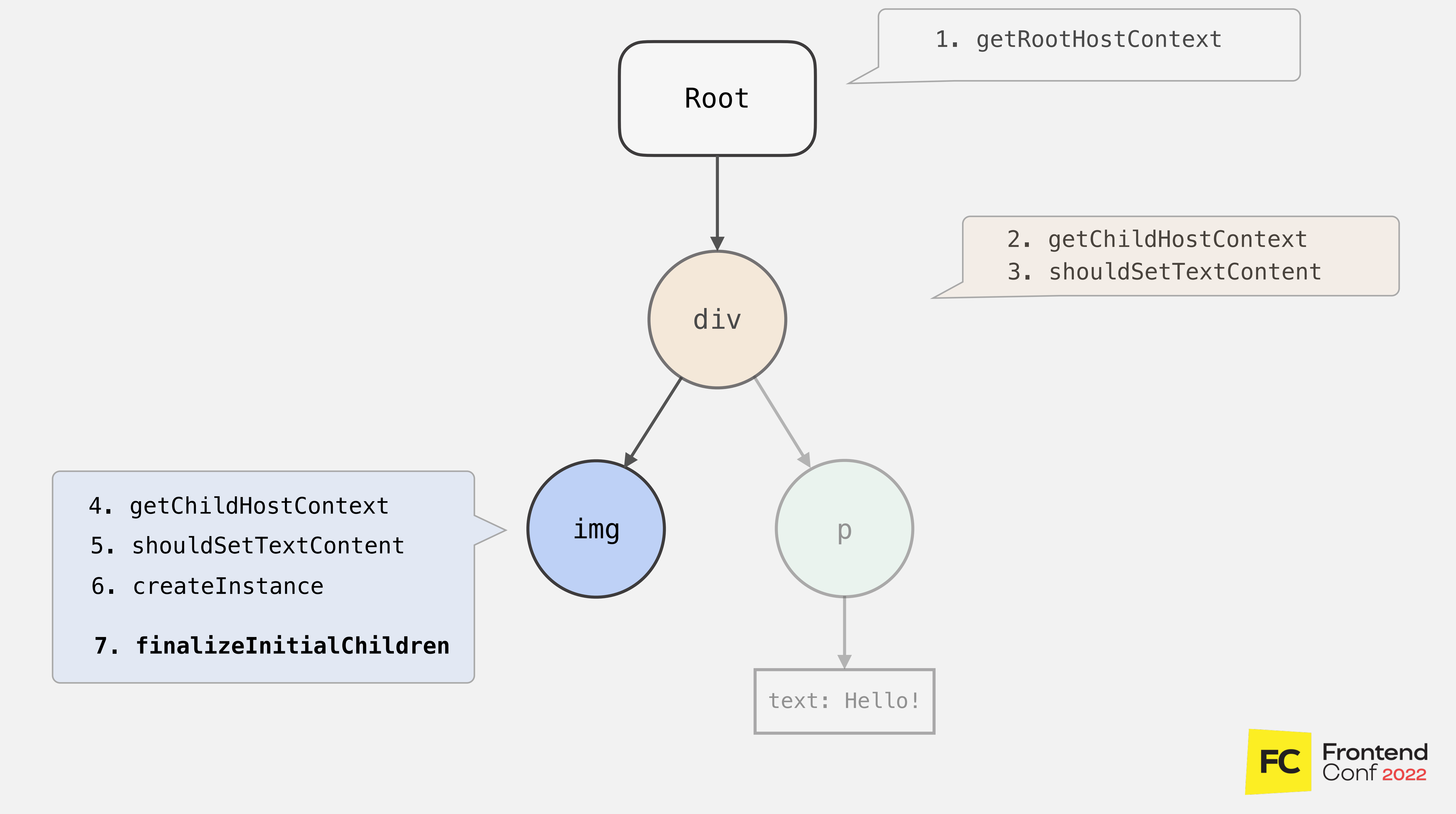
```
createInstance: (type, props) => {
    const instance = document.createElement(type);
    if (props.className) {
        instance.className = props.className;
    }

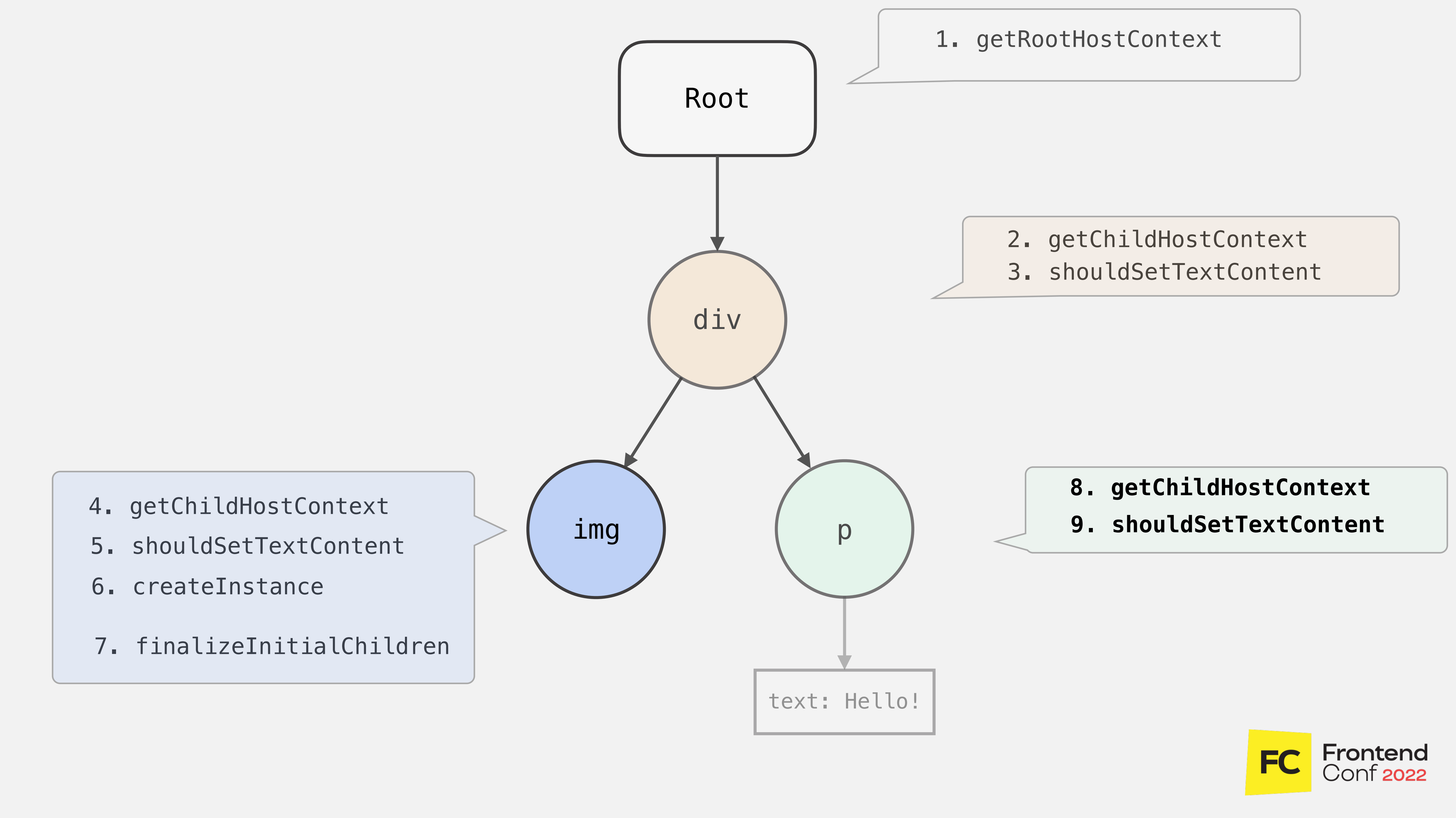
    if (instance.tagName === 'IMG' && props.src) {
        instance.src = props.src;
    }

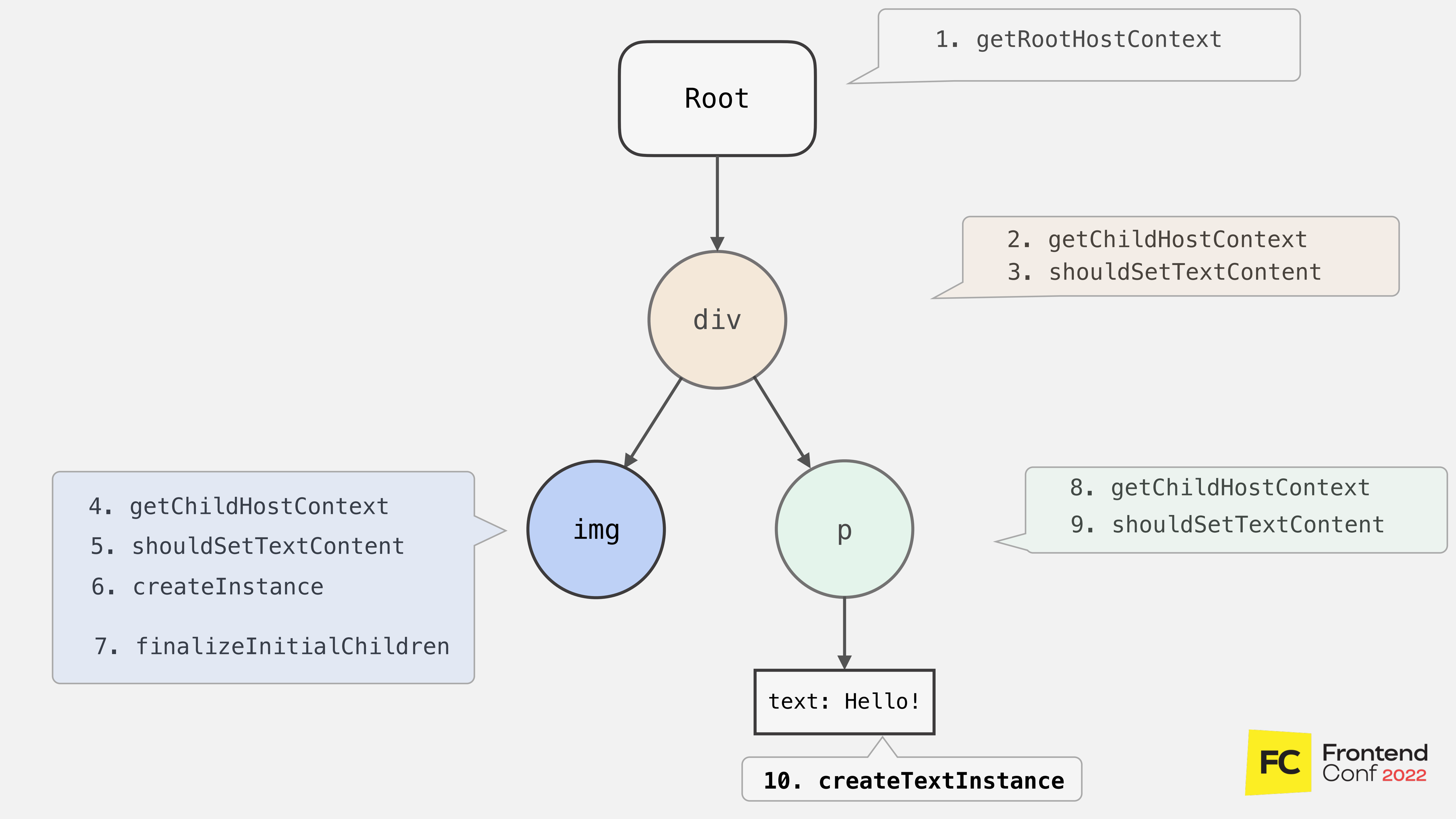
    return instance;
}
```

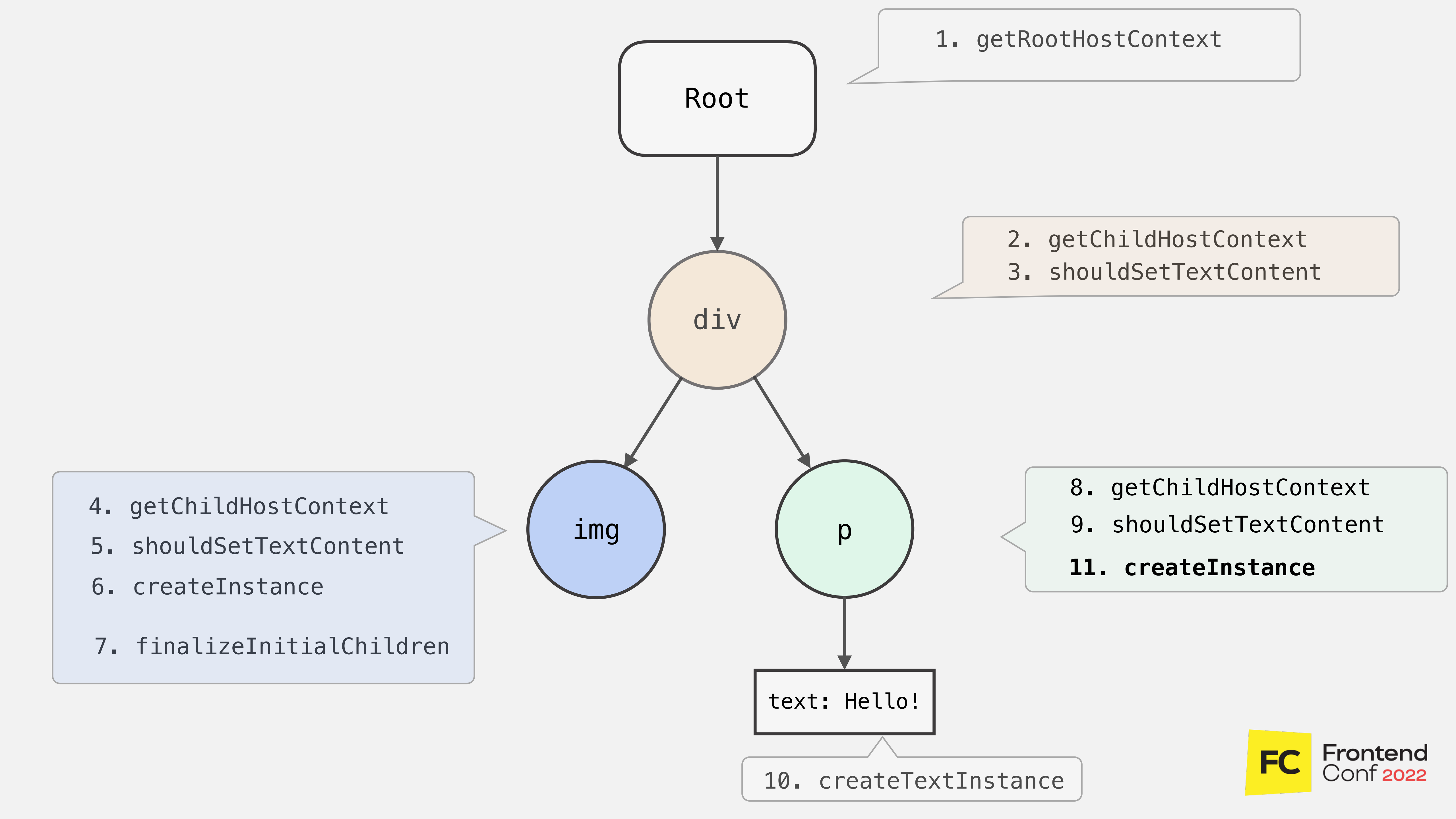




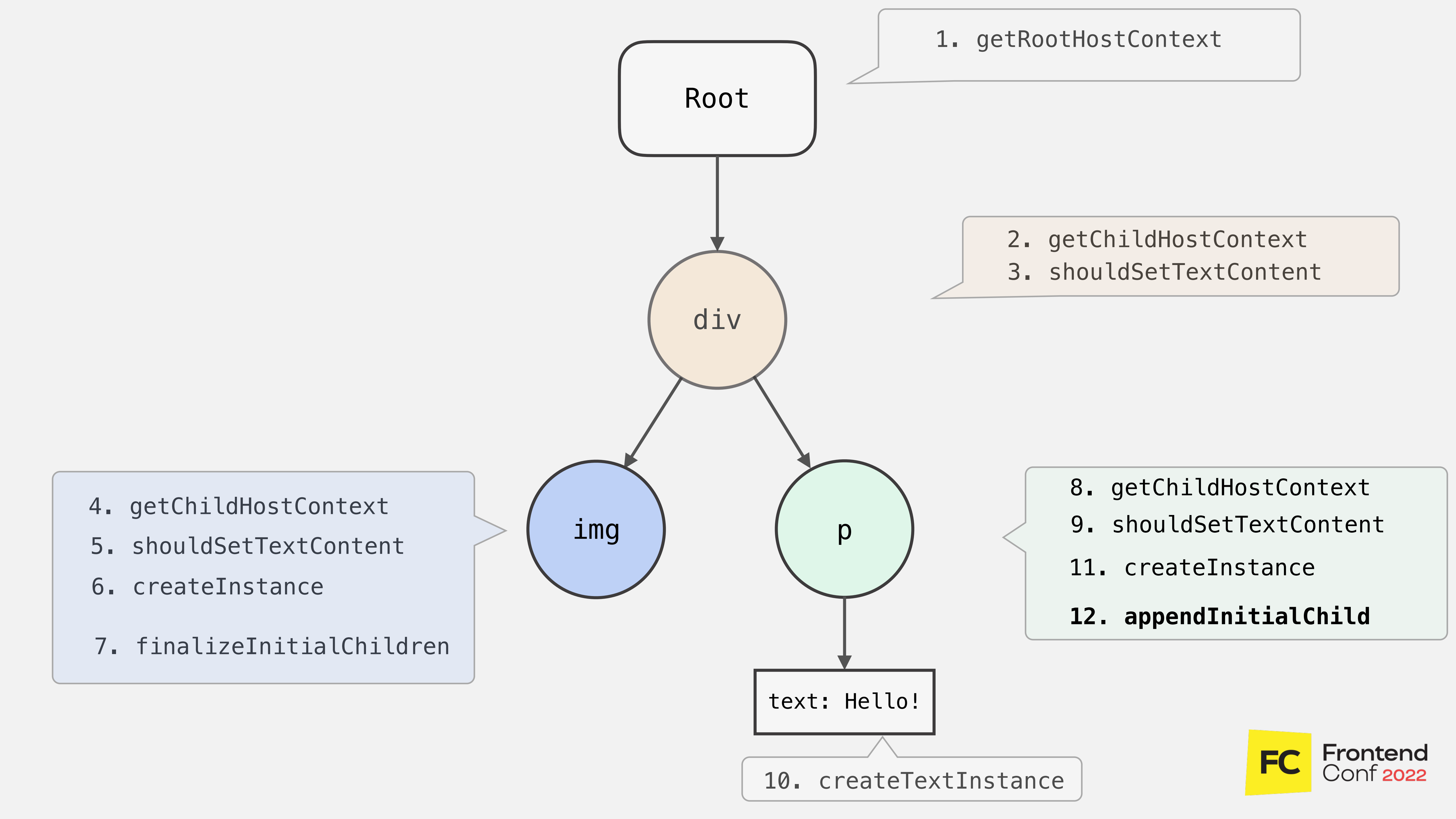

1. getRootHostContext
Root
2. getChildHostContext
3. shouldSetTextContent
div
4. getChildHostContext
5. shouldSetTextContent
6. createInstance
7. finalizeInitialChildren
img
p
text: Hello!
FC
Frontend
Conf 2022


1. getRootHostContext
Root
2. getChildHostContext
3. shouldSetTextContent
div
img
p
4. getChildHostContext
5. shouldSetTextContent
6. createInstance
7. finalizeInitialChildren
8. getChildHostContext
9. shouldSetTextContent
text: Hello!
FC Frontend Conf 2022


Root
1. getRootHostContext
div
2. getChildHostContext
3. shouldSetTextContent
img
4. getChildHostContext
5. shouldSetTextContent
6. createInstance
7. finalizeInitialChildren
p
8. getChildHostContext
9. shouldSetTextContent
text: Hello!
10. createTextInstance
FC
Frontend
Conf 2022



Root
1. getRootHostContext
div
2. getChildHostContext
3. shouldSetTextContent
img
p
4. getChildHostContext
5. shouldSetTextContent
6. createInstance
7. finalizeInitialChildren
8. getChildHostContext
9. shouldSetTextContent
11. createInstance
text: Hello!
10. createTextInstance
FC
Frontend
Conf 2022

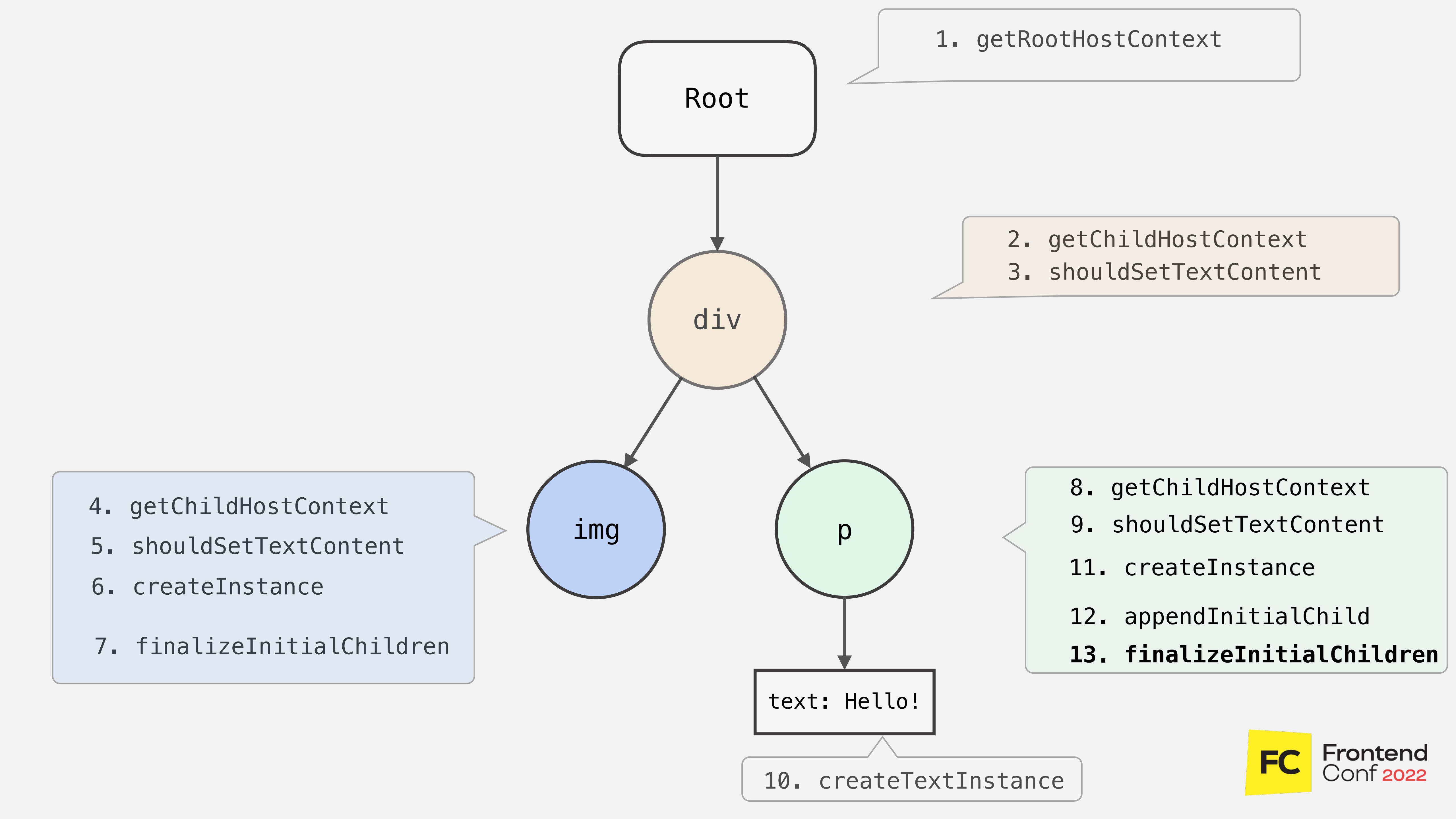

1. getRootHostContext
Root
2. getChildHostContext
3. shouldSetTextContent
div
img
p
4. getChildHostContext
5. shouldSetTextContent
6. createInstance
7. finalizeInitialChildren
8. getChildHostContext
9. shouldSetTextContent
11. createInstance
12. appendInitialChild
text: Hello!
10. createTextInstance
FC
Frontend
Conf 2022

hostConfig

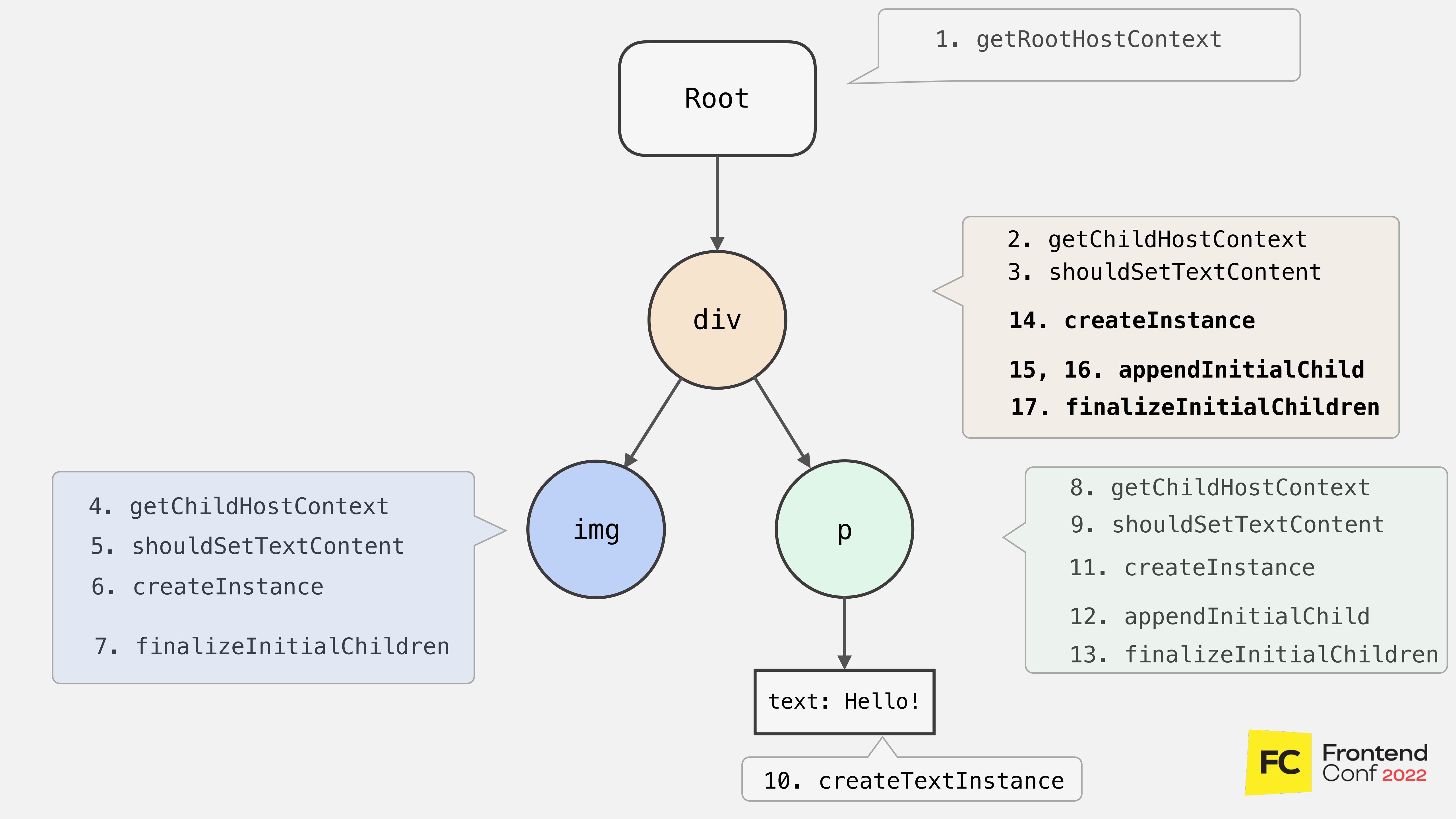
```
appendInitialChild: (parent, child) => {
    parent.appendChild(child);
}
```

- Прикрепляет ребёнка к родителю





Root
1. getRootHostContext
div
2. getChildHostContext
3. shouldSetTextContent
img
4. getChildHostContext
5. shouldSetTextContent
6. createInstance
7. finalizeInitialChildren
p
8. getChildHostContext
9. shouldSetTextContent
11. createInstance
12. appendInitialChild
13. finalizeInitialChildren
text: Hello!
10. createTextInstance
FC
Frontend
Conf 2022



1. getRootHostContext
Root
div
2. getChildHostContext
3. shouldSetTextContent
14. createInstance
15, 16. appendInitialChild
17. finalizeInitialChildren
img
p
4. getChildHostContext
5. shouldSetTextContent
6. createInstance
7. finalizeInitialChildren
8. getChildHostContext
9. shouldSetTextContent
11. createInstance
12. appendInitialChild
13. finalizeInitialChildren
text: Hello!
10. createTextInstance
FC
Frontend
Conf 2022

setState

Diff & Patch

Layout

Paint

Render Phase

Commit Phase



setState

Diff & Patch

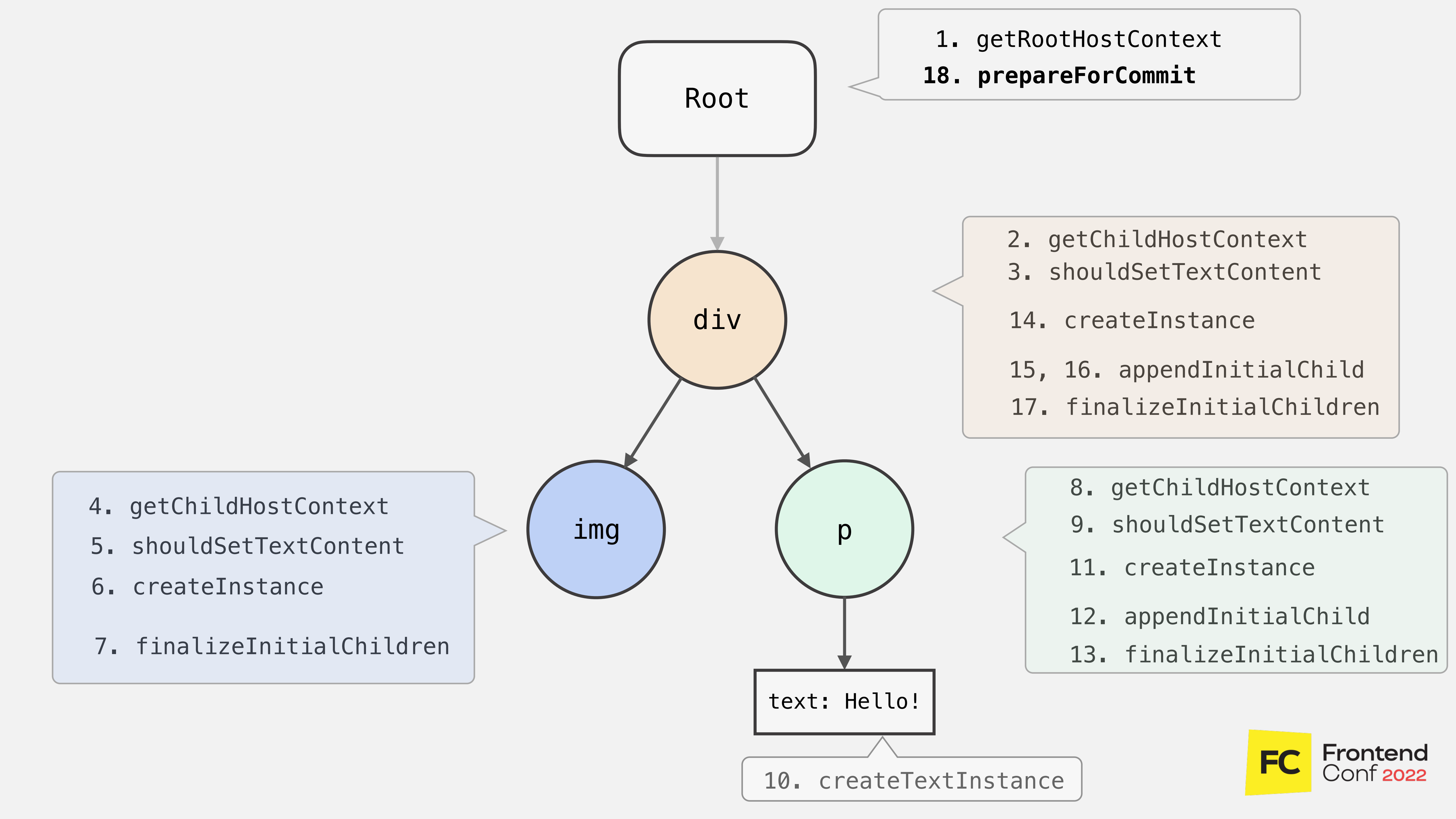
Layout

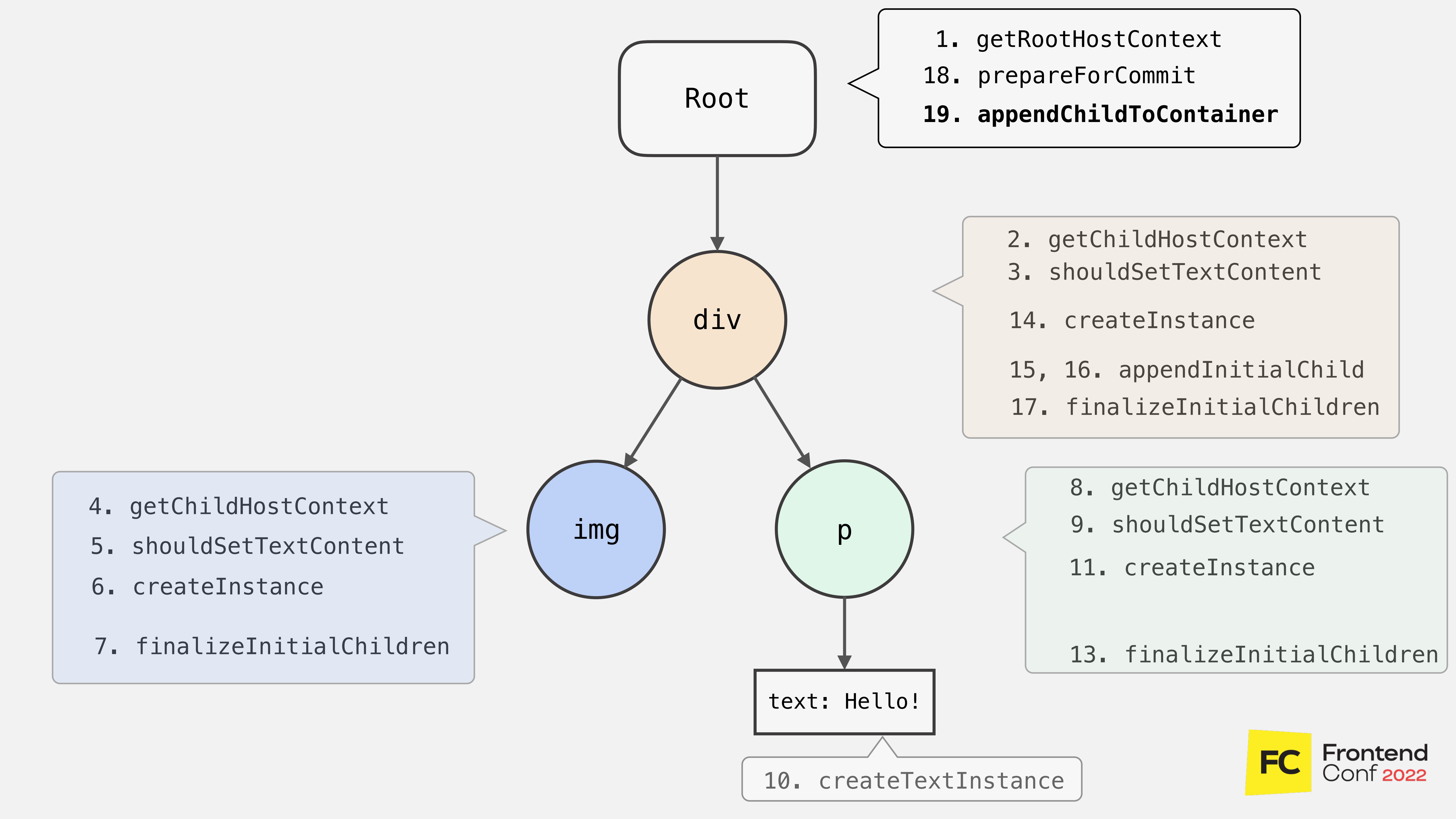
Paint

Render Phase

Commit Phase




Root
1. getRootHostContext
18. prepareForCommit
div
2. getChildHostContext
3. shouldSetTextContent
14. createInstance
15, 16. appendInitialChild
17. finalizeInitialChildren
img
4. getChildHostContext
5. shouldSetTextContent
6. createInstance
7. finalizeInitialChildren
p
8. getChildHostContext
9. shouldSetTextContent
11. createInstance
12. appendInitialChild
13. finalizeInitialChildren
text: Hello!
10. createTextInstance
FC
Frontend
Conf 2022



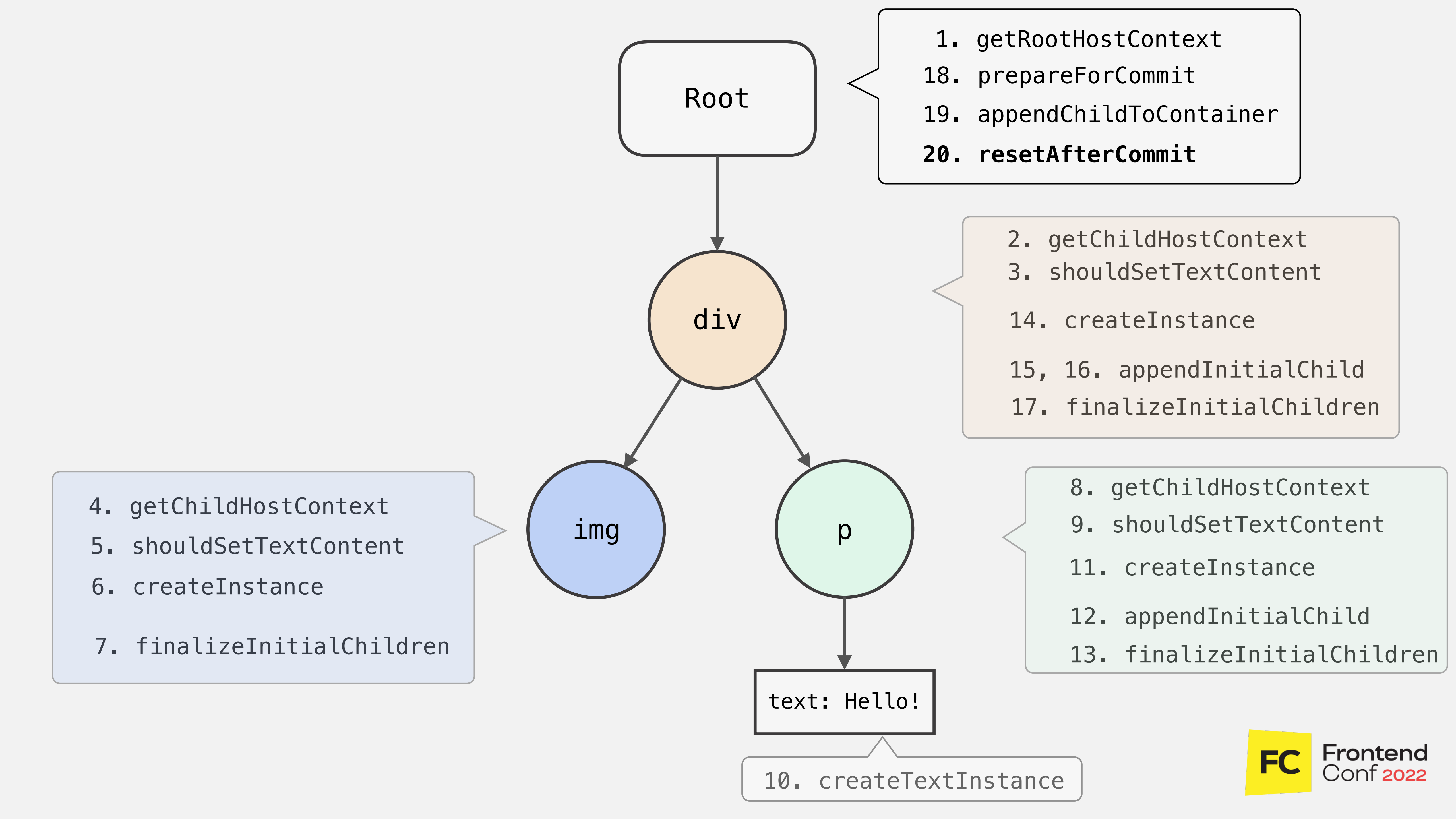
Root
1. getRootHostContext
18. prepareForCommit
19. appendChildToContainer
div
2. getChildHostContext
3. shouldSetTextContent
14. createInstance
15, 16. appendInitialChild
17. finalizeInitialChildren
img
4. getChildHostContext
5. shouldSetTextContent
6. createInstance
7. finalizeInitialChildren
p
8. getChildHostContext
9. shouldSetTextContent
11. createInstance
13. finalizeInitialChildren
text: Hello!
10. createTextInstance
FC
Frontend
Conf 2022

hostConfig

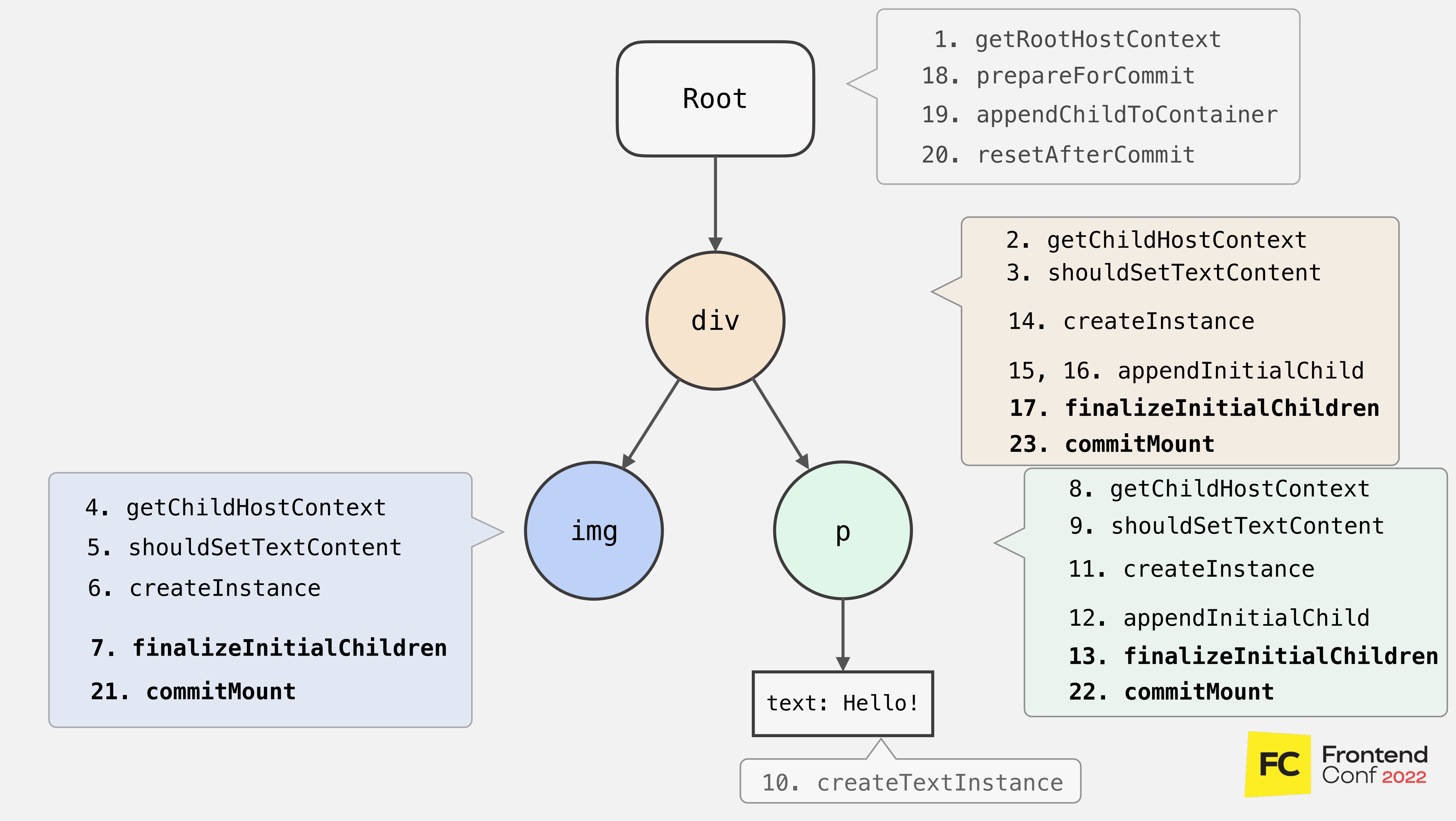
```
appendChildToContainer: (container, child) => {
    container.appendChild(child);
}
```

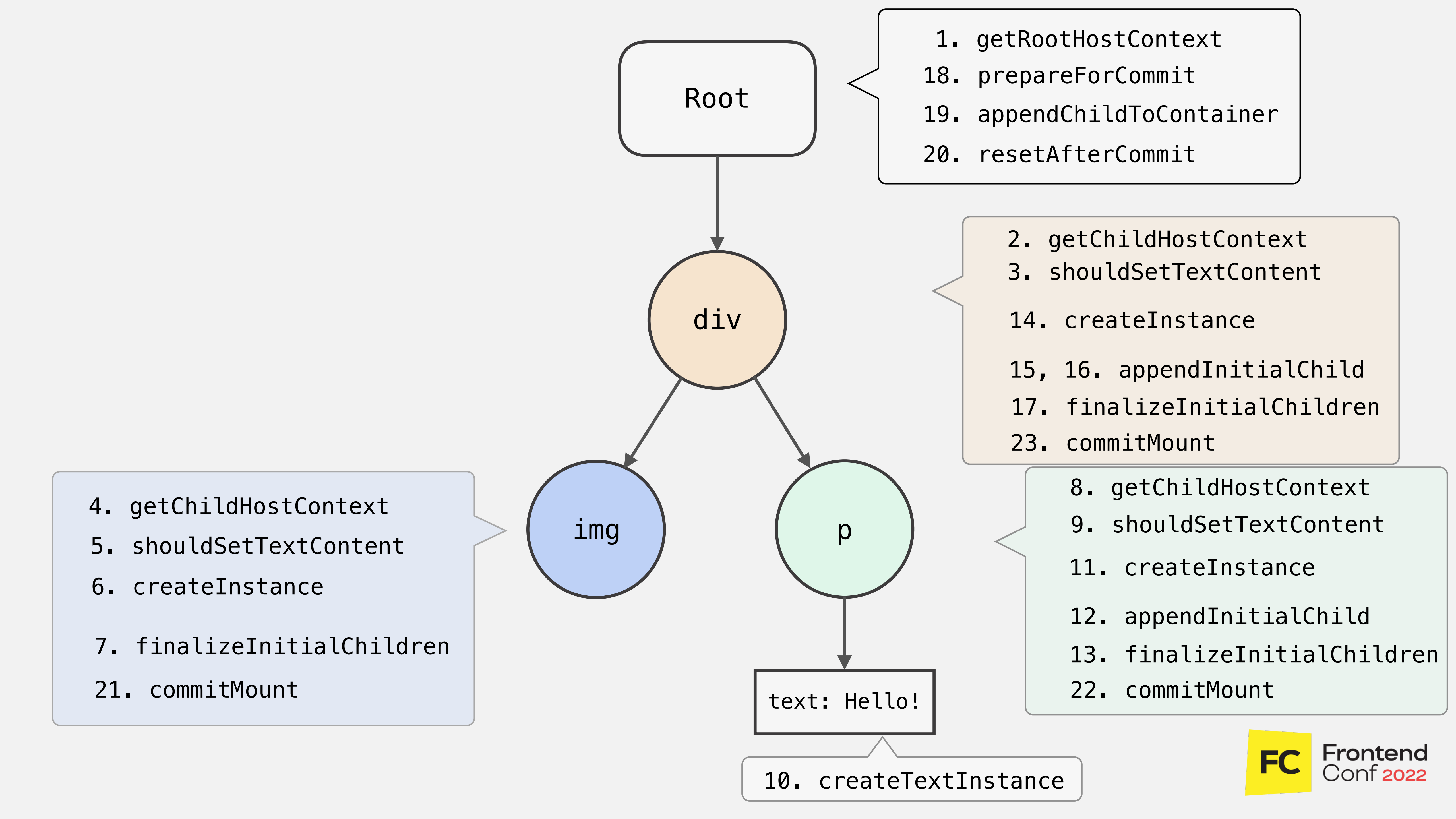
- Добавляет ребёнка к корневому контейнеру
- Хранящееся в памяти дерево отрисовывается на экран

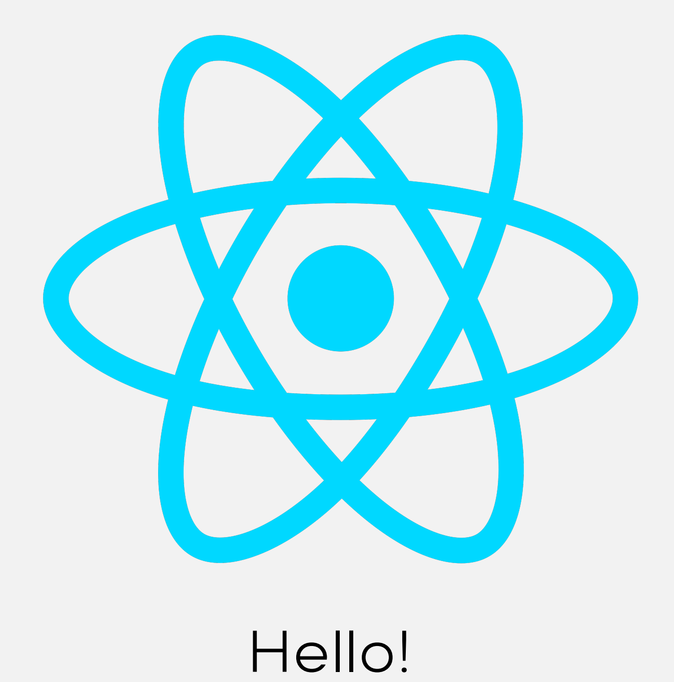




Root
1. getRootHostContext
18. prepareForCommit
19. appendChildToContainer
20. resetAfterCommit
div
2. getChildHostContext
3. shouldSetTextContent
14. createInstance
15, 16. appendInitialChild
17. finalizeInitialChildren
img
4. getChildHostContext
5. shouldSetTextContent
6. createInstance
7. finalizeInitialChildren
p
8. getChildHostContext
9. shouldSetTextContent
11. createInstance
12. appendInitialChild
13. finalizeInitialChildren
text: Hello!
10. createTextInstance
FC
Frontend Conf 2022


Root
1. getRootHostContext
18. prepareForCommit
19. appendChildToContainer
20. resetAfterCommit
div
2. getChildHostContext
3. shouldSetTextContent
14. createInstance
15, 16. appendInitialChild
17. finalizeInitialChildren
23. commitMount
img
4. getChildHostContext
5. shouldSetTextContent
6. createInstance
7. finalizeInitialChildren
21. commitMount
p
8. getChildHostContext
9. shouldSetTextContent
11. createInstance
12. appendInitialChild
13. finalizeInitialChildren
22. commitMount
text: Hello!
10. createTextInstance
FC Frontend Conf 2022


Root
1. getRootHostContext
18. prepareForCommit
19. appendChildToContainer
20. resetAfterCommit
div
2. getChildHostContext
3. shouldSetTextContent
14. createInstance
15, 16. appendInitialChild
17. finalizeInitialChildren
23. commitMount
img
4. getChildHostContext
5. shouldSetTextContent
6. createInstance
7. finalizeInitialChildren
21. commitMount
p
8. getChildHostContext
9. shouldSetTextContent
11. createInstance
12. appendInitialChild
13. finalizeInitialChildren
22. commitMount
text: Hello!
10. createTextInstance
FC
Frontend
Conf 2022



Hello!



Разработка своего движка




PixiJS


FC
Frontend
Conf 2022


PixiJS


FC
Frontend
Conf 2022



- Opensource-библиотека с большим комьюнити





- Opensource-библиотека с большим комьюнити
- **Рендеринг в 2D webGL**





- Opensource-библиотека с большим комьюнити
- Рендеринг в 2D webGL
- **API для работы с интерактивной графикой**



```
const app = new PIXI.Application({
   width: 800, height: 600,
   backgroundColor: 0x1099bb,
   resolution: window.devicePixelRatio || 1,
});

document.body.appendChild(app.view);

const container = new PIXI.Container();

app.stage.addChild(container);

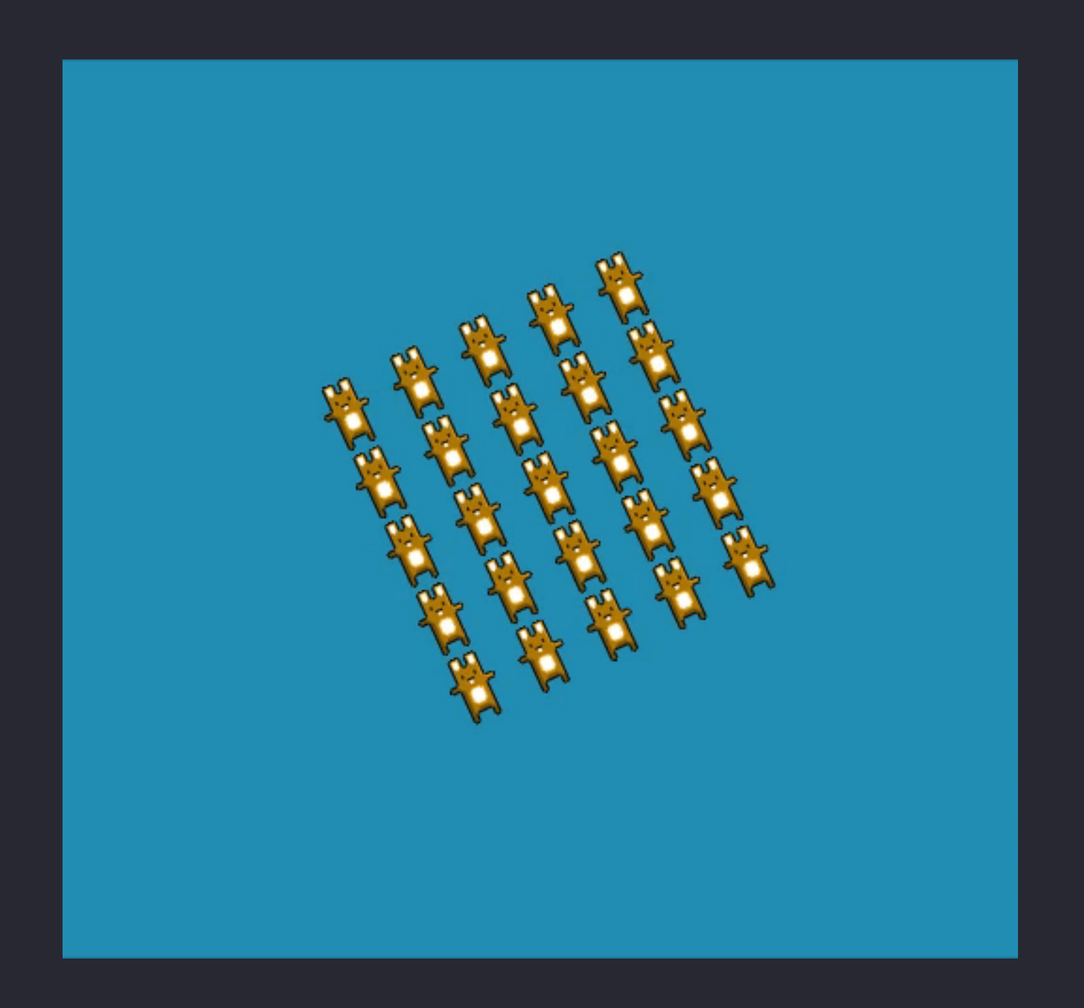
// Create a new texture
const texture = PIXI.Texture.from('examples/
assets/bunny.png');

// Create a 5x5 grid of bunnies
for (let i = 0; i < 25; i++) {
    const bunny = new PIXI.Sprite(texture);
    bunny.anchor.set(0.5);
    bunny.x = (i % 5) * 40;
    bunny.y = Math.floor(i / 5) * 40;
    container.addChild(bunny);
}

container.x = app.screen.width / 2;
container.y = app.screen.height / 2;

container.pivot.x = container.width / 2;
container.pivot.y = container.height / 2;

app.ticker.add((delta) => {
    container.rotation -= 0.01 * delta;
});
```





Создаём инстанс PIXI.Application

```
const app = new PIXI.Application({
   width: 800, height: 600,
   backgroundColor: 0x1099bb,
   resolution: window.devicePixelRatio || 1,
});

document.body.appendChild(app.view);

const container = new PIXI.Container();

app.stage.addChild(container);

// Create a new texture
const texture = PIXI.Texture.from('examples/
assets/bunny.png');

// Create a 5x5 grid of bunnies
for (let i = 0; i < 25; i++) {
    const bunny = new PIXI.Sprite(texture);
    bunny.anchor.set(0.5);
    bunny.x = (i % 5) * 40;
    bunny.y = Math.floor(i / 5) * 40;
    container.addChild(bunny);
}

container.x = app.screen.width / 2;
container.y = app.screen.height / 2;

container.pivot.x = container.width / 2;
container.pivot.y = container.height / 2;

app.ticker.add((delta) => {
    container.rotation -= 0.01 * delta;
});
```



```
const app = new PIXI.Application({
  width: 800, height: 600,
  backgroundColor: 0x1099bb,
  resolution: window.devicePixelRatio || 1,
});

document.body.appendChild(app.view);

const container = new PIXI.Container();

app.stage.addChild(container);

// Create a new texture
const texture = PIXI.Texture.from('examples/
assets/bunny.png');

// Create a 5x5 grid of bunnies
for (let i = 0; i < 25; i++) {
    const bunny = new PIXI.Sprite(texture);
    bunny.anchor.set(0.5);
    bunny.x = (i % 5) * 40;
    bunny.y = Math.floor(i / 5) * 40;
    container.addChild(bunny);
}

container.x = app.screen.width / 2;
container.y = app.screen.height / 2;

container.pivot.x = container.width / 2;
container.pivot.y = container.height / 2;

app.ticker.add((delta) => {
    container.rotation -= 0.01 * delta;
});
```

Создаём инстанс PIXI.Application

Прикрепляем к веб-странице



```
const app = new PIXI.Application({
   width: 800, height: 600,
   backgroundColor: 0x1099bb,
   resolution: window.devicePixelRatio || 1,
});

document.body.appendChild(app.view);

const container = new PIXI.Container();

app.stage.addChild(container);

// Create a new texture
const texture = PIXI.Texture.from('examples/
assets/bunny.png');

// Create a 5x5 grid of bunnies
for (let i = 0; i < 25; i++) {
    const bunny = new PIXI.Sprite(texture);
    bunny.anchor.set(0.5);
    bunny.x = (i % 5) * 40;
    bunny.y = Math.floor(i / 5) * 40;
    container.addChild(bunny);
}

container.x = app.screen.width / 2;
container.y = app.screen.height / 2;

container.pivot.x = container.width / 2;
container.pivot.y = container.height / 2;

app.ticker.add((delta) => {
    container.rotation -= 0.01 * delta;
});
```

Создаём инстанс PIXI.Application

Прикрепляем к веб-странице

Создаём инстанс Container



```
const app = new PIXI.Application({
   width: 800, height: 600,
   backgroundColor: 0x1099bb,
   resolution: window.devicePixelRatio || 1,
});

document.body.appendChild(app.view);

const container = new PIXI.Container();

app.stage.addChild(container);

// Create a new texture
const texture = PIXI.Texture.from('examples/
assets/bunny.png');

// Create a 5x5 grid of bunnies
for (let i = 0; i < 25; i++) {
    const bunny = new PIXI.Sprite(texture);
    bunny.anchor.set(0.5);
    bunny.x = (i % 5) * 40;
    bunny.y = Math.floor(i / 5) * 40;
    container.addChild(bunny);
}

container.x = app.screen.width / 2;
container.y = app.screen.height / 2;

container.pivot.x = container.width / 2;
container.pivot.y = container.height / 2;

app.ticker.add((delta) => {
    container.rotation -= 0.01 * delta;
});
```

Создаём инстанс PIXI.Application

Прикрепляем к веб-странице

Создаём инстанс Container

Прикрепляем контейнер к «root»



```
const app = new PIXI.Application({
   width: 800, height: 600,
   backgroundColor: 0x1099bb,
   resolution: window.devicePixelRatio || 1,
});

document.body.appendChild(app.view);

const container = new PIXI.Container();

app.stage.addChild(container);

// Create a new texture
const texture = PIXI.Texture.from('examples/
assets/bunny.png');

// Create a 5x5 grid of bunnies
for (let i = 0; i < 25; i++) {
    const bunny = new PIXI.Sprite(texture);
    bunny.anchor.set(0.5);
    bunny.x = (i % 5) * 40;
    bunny.y = Math.floor(i / 5) * 40;
    container.addChild(bunny);
}

container.x = app.screen.width / 2;
container.y = app.screen.height / 2;

container.pivot.x = container.width / 2;
container.pivot.y = container.height / 2;

app.ticker.add((delta) => {
    container.rotation -= 0.01 * delta;
});
```

Создаём инстанс PIXI.Application

Прикрепляем к веб-странице

Создаём инстанс Container

Прикрепляем контейнер к «root»

Создаём инстанс Texture



```
const app = new PIXI.Application({
   width: 800, height: 600,
   backgroundColor: 0x1099bb,
   resolution: window.devicePixelRatio || 1,
});

document.body.appendChild(app.view);

const container = new PIXI.Container();

app.stage.addChild(container);

// Create a new texture
const texture = PIXI.Texture.from('examples/
assets/bunny.png');

// Create a 5x5 grid of bunnies
for (let i = 0; i < 25; i++) {
    const bunny = new PIXI.Sprite(texture);
    bunny.anchor.set(0.5);
    bunny.x = (i % 5) * 40;
    bunny.y = Math.floor(i / 5) * 40;
    container.addChild(bunny);
}

container.x = app.screen.width / 2;
container.y = app.screen.height / 2;

container.pivot.x = container.width / 2;
container.pivot.y = container.height / 2;

app.ticker.add((delta) => {
    container.rotation -= 0.01 * delta;
});
```

Создаём инстанс PIXI.Application

Прикрепляем к веб-странице

Создаём инстанс Container

Прикрепляем контейнер к «root»

Создаём инстанс Texture

Создаём инстанс Sprite



```
const app = new PIXI.Application({
  width: 800, height: 600,
  backgroundColor: 0x1099bb,
  resolution: window.devicePixelRatio || 1,
});

document.body.appendChild(app.view);

const container = new PIXI.Container();

app.stage.addChild(container);

// Create a new texture
const texture = PIXI.Texture.from('examples/
assets/bunny.png');

// Create a 5x5 grid of bunnies
for (let i = 0; i < 25; i++) {
    const bunny = new PIXI.Sprite(texture);
    bunny.anchor.set(0.5);
    bunny.x = (i % 5) * 40;
    bunny.y = Math.floor(i / 5) * 40;
    container.addChild(bunny);
}

container.x = app.screen.width / 2;
container.y = app.screen.height / 2;

container.pivot.x = container.width / 2;
container.pivot.y = container.height / 2;

app.ticker.add((delta) => {
    container.rotation -= 0.01 * delta;
});
```

Создаём инстанс PIXI.Application

Прикрепляем к веб-странице

Создаём инстанс Container

Прикрепляем контейнер к «root»

Создаём инстанс Texture

Создаём инстанс Sprite

Задаём свойства (props) для sprite



```js
const app = new PIXI.Application({
   width: 800, height: 600,
   backgroundColor: 0x1099bb,
   resolution: window.devicePixelRatio || 1,
});

document.body.appendChild(app.view);

const container = new PIXI.Container();

app.stage.addChild(container);

// Create a new texture
const texture = PIXI.Texture.from('examples/
assets/bunny.png');

// Create a 5x5 grid of bunnies
for (let i = 0; i < 25; i++) {
    const bunny = new PIXI.Sprite(texture);
    bunny.anchor.set(0.5);
    bunny.x = (i % 5) * 40;
    bunny.y = Math.floor(i / 5) * 40;
    container.addChild(bunny);
}

container.x = app.screen.width / 2;
container.y = app.screen.height / 2;

container.pivot.x = container.width / 2;
container.pivot.y = container.height / 2;

app.ticker.add((delta) => {
    container.rotation -= 0.01 * delta;
});
```

Создаём инстанс PIXI.Application

Прикрепляем к веб-странице

Создаём инстанс Container

Прикрепляем контейнер к «root»

Создаём инстанс Texture

Создаём инстанс Sprite

Задаём свойства (props) для sprite

Добавляем в дерево sprite



```
const app = new PIXI.Application({
   width: 800, height: 600,
   backgroundColor: 0x1099bb,
   resolution: window.devicePixelRatio || 1,
});

document.body.appendChild(app.view);

const container = new PIXI.Container();

app.stage.addChild(container);

// Create a new texture
const texture = PIXI.Texture.from('examples/
assets/bunny.png');

// Create a 5x5 grid of bunnies
for (let i = 0; i < 25; i++) {
    const bunny = new PIXI.Sprite(texture);
    bunny.anchor.set(0.5);
    bunny.x = (i % 5) * 40;
    bunny.y = Math.floor(i / 5) * 40;
    container.addChild(bunny);
}

container.x = app.screen.width / 2;
container.y = app.screen.height / 2;

container.pivot.x = container.width / 2;
container.pivot.y = container.height / 2;

app.ticker.add((delta) => {
    container.rotation -= 0.01 * delta;
});
```

Создаём инстанс PIXI.Application

Прикрепляем к веб-странице

Создаём инстанс Container

Прикрепляем контейнер к «root»

Создаём инстанс Texture

Создаём инстанс Sprite

Задаём свойства (props) для sprite

Добавляем в дерево sprite

Задаём свойства (props) для container



```
const app = new PIXI.Application({
  width: 800, height: 600,
  backgroundColor: 0x1099bb,
  resolution: window.devicePixelRatio || 1,
});

document.body.appendChild(app.view);

const container = new PIXI.Container();

app.stage.addChild(container);

// Create a new texture
const texture = PIXI.Texture.from('examples/
assets/bunny.png');

// Create a 5x5 grid of bunnies
for (let i = 0; i < 25; i++) {
    const bunny = new PIXI.Sprite(texture);
    bunny.anchor.set(0.5);
    bunny.x = (i % 5) * 40;
    bunny.y = Math.floor(i / 5) * 40;
    container.addChild(bunny);
}

container.x = app.screen.width / 2;
container.y = app.screen.height / 2;

container.pivot.x = container.width / 2;
container.pivot.y = container.height / 2;

app.ticker.add((delta) => {
    container.rotation -= 0.01 * delta;
});
```

Создаём инстанс PIXI.Application

Прикрепляем к веб-странице

Создаём инстанс Container

Прикрепляем контейнер к «root»

Создаём инстанс Texture

Создаём инстанс Sprite

Задаём свойства (props) для sprite

Добавляем в дерево sprite

Задаём свойства (props) для container

Хук для анимации



```
const app = new PIXI.Application({
   width: 800, height: 600,
   backgroundColor: 0x1099bb,
   resolution: window.devicePixelRatio || 1,
});

document.body.appendChild(app.view);

const container = new PIXI.Container();

app.stage.addChild(container);

// Create a new texture
const texture = PIXI.Texture.from('examples/
assets/bunny.png');

// Create a 5x5 grid of bunnies
for (let i = 0; i < 25; i++) {
    const bunny = new PIXI.Sprite(texture);
    bunny.anchor.set(0.5);
    bunny.x = (i % 5) * 40;
    bunny.y = Math.floor(i / 5) * 40;
    container.addChild(bunny);
}

container.x = app.screen.width / 2;
container.y = app.screen.height / 2;

container.pivot.x = container.width / 2;
container.pivot.y = container.height / 2;

app.ticker.add((delta) => {
    container.rotation -= 0.01 * delta;
});
```

Создаём инстанс PIXI.Application

Прикрепляем к веб-странице

Создаём инстанс Container

Прикрепляем контейнер к «root»

Создаём инстанс Texture

Создаём инстанс Sprite

Задаём свойства (props) для sprite

Добавляем в дерево sprite

Задаём свойства (props) для container

Хук для анимации



```
const app = new PIXI.Application({
  width: 800, height: 600,
  backgroundColor: 0x1099bb,
  resolution: window.devicePixelRatio || 1,
});

document.body.appendChild(app.view);

const container = new PIXI.Container();

app.stage.addChild(container);

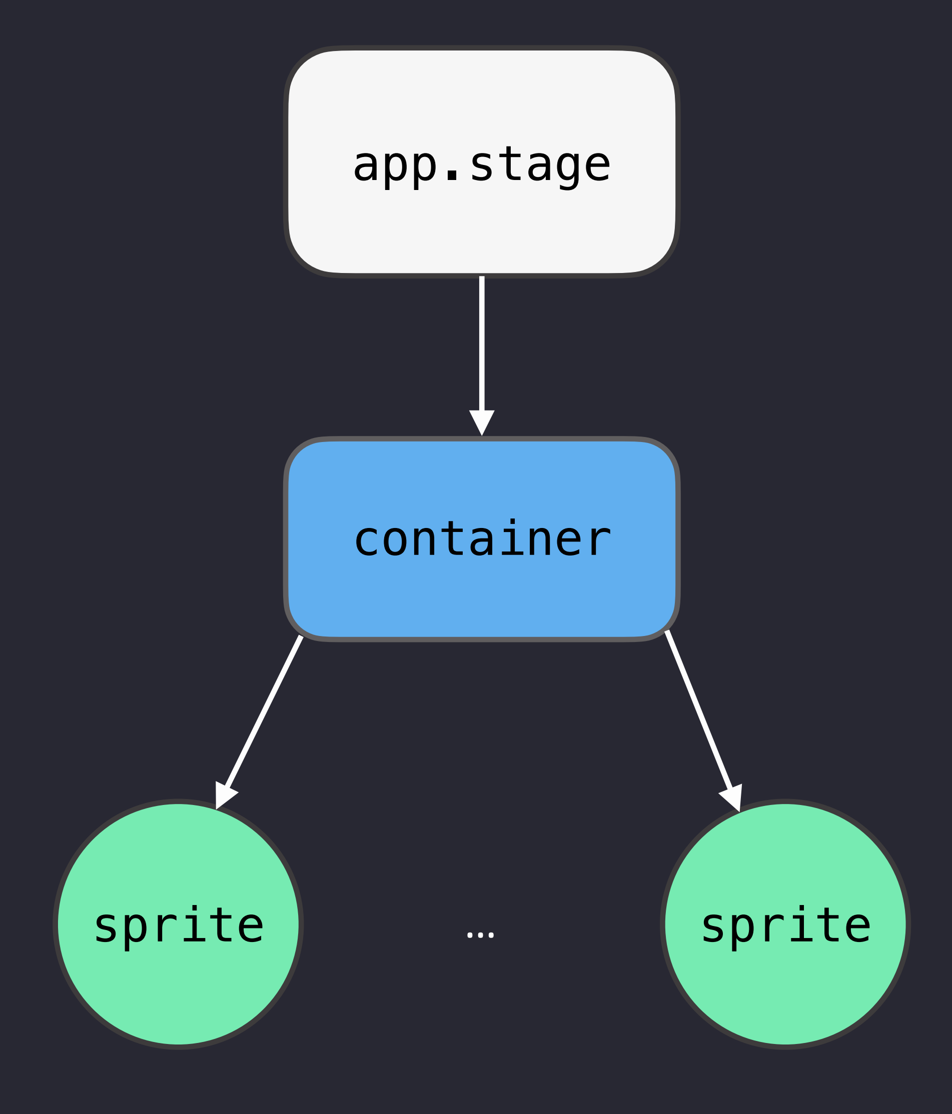
// Create a new texture
const texture = PIXI.Texture.from('examples/assets/bunny.png');

// Create a 5x5 grid of bunnies
for (let i = 0; i < 25; i++) {
    const bunny = new PIXI.Sprite(texture);
    bunny.anchor.set(0.5);
    bunny.x = (i % 5) * 40;
    bunny.y = Math.floor(i / 5) * 40;
    container.addChild(bunny);
}

container.x = app.screen.width / 2;
container.y = app.screen.height / 2;

container.pivot.x = container.width / 2;
container.pivot.y = container.height / 2;

app.ticker.add((delta) => {
    container.rotation -= 0.01 * delta;
});
```





React PIXI renderer



React PIXI renderer

```
import React from "react";
import { render } from "./fiber/renderer";
import * as PIXI from "pixi.js";
import sprite from "./assets/sprite.png";

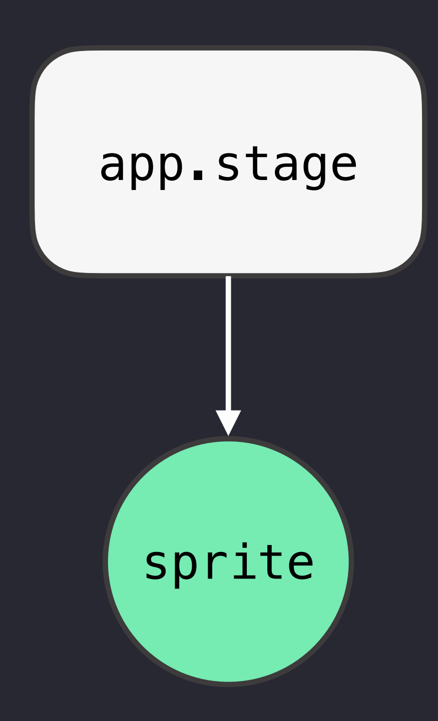
const canvas = document.getElementById("canvas");

const app = new PIXI.Application({
  width: 800,
  height: 600,
  view: canvas,
  backgroundColor: 0x292c33,
});

render(<App />, app.stage);

const texture = PIXI.Texture.from(sprite);

function App() {
  return <sprite texture={texture} width={100} height={100} />;
}
```





hostConfig

```
createInstance: (type, props) => {
      const instance = new PIXI.Sprite(props.texture);

      instance.width = props.width;
      instance.height = props.height;

    return instance;
  }
```



hostConfig

```
appendInitialChild: (parent, child) => {
    parent.addChild(child);
  },
```



hostConfig

```
appendChildToContainer: (container, child) => {
    container.addChild(child);
  },
```






FC
Frontend
Conf 2022

Удалить и добавить

```
const canvas = document.getElementById("canvas");

const app = new PIXI.Application({
  width: 800,
  height: 600,
  view: canvas,
  backgroundColor: 0x292c33,
});

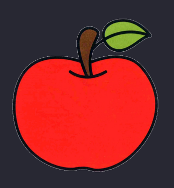
render(<App />, app.stage);

const boy = PIXI.Texture.from(boyImg);
const apple = PIXI.Texture.from(appleImg);

function App() {
  const [visible, setVisible] = useState(true);

  const handleClick = () => {
    setVisible(false);
  };

  return (
    <>
      {visible && <sprite texture={apple} width={100} height={75} x={650} y={250} />}
       <sprite texture={boy} width={200} height={170} x={50} y={200} buttonMode onClick={handleClick} />
    </>
  );
}
```





Обработка событий

```
createInstance: (type, props) => {
  const { x = 0, y = 0 } = props;

  const instance = new PIXI.Sprite(props.texture);

  instance.width = props.width;
  instance.height = props.height;

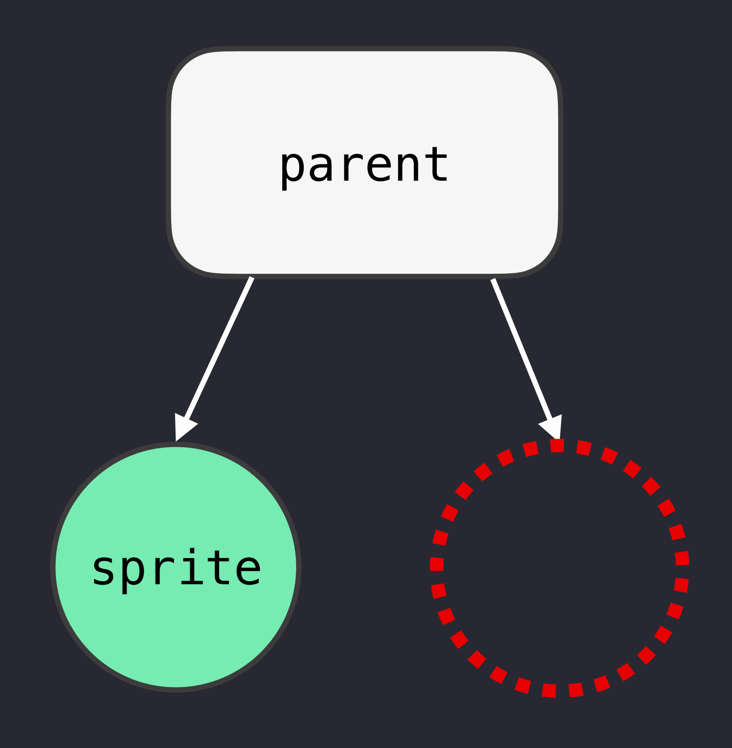
  instance.x = x;
  instance.y = y;

  if (props.onClick) {
    instance.interactive = true;
    instance.on('click', props.onClick);
  }
  return instance;
},
```



Удаление

```
removeChild: (parentInstance, child) => {
  parentInstance.removeChild(child);
},
```

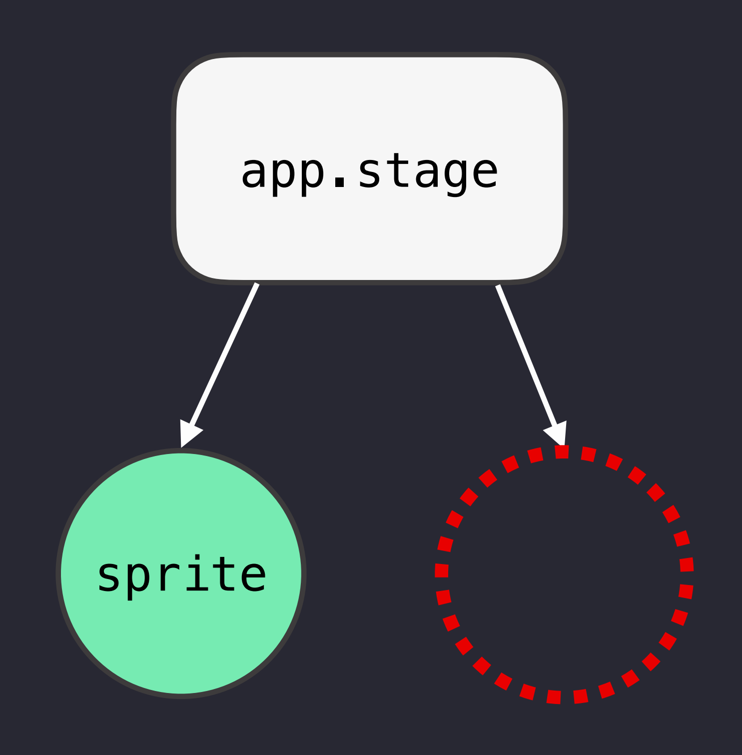


Вызывается во время коммит-фазы для родителя, поддерево которого должно быть удалено



Удаление

```
removeChildFromContainer: (container, child) => {
    container.removeChild(child);
  },
```

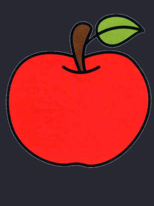


Вызывается во время коммит-фазы для корневого контейнера, поддерево которого должно быть удалено



Демо





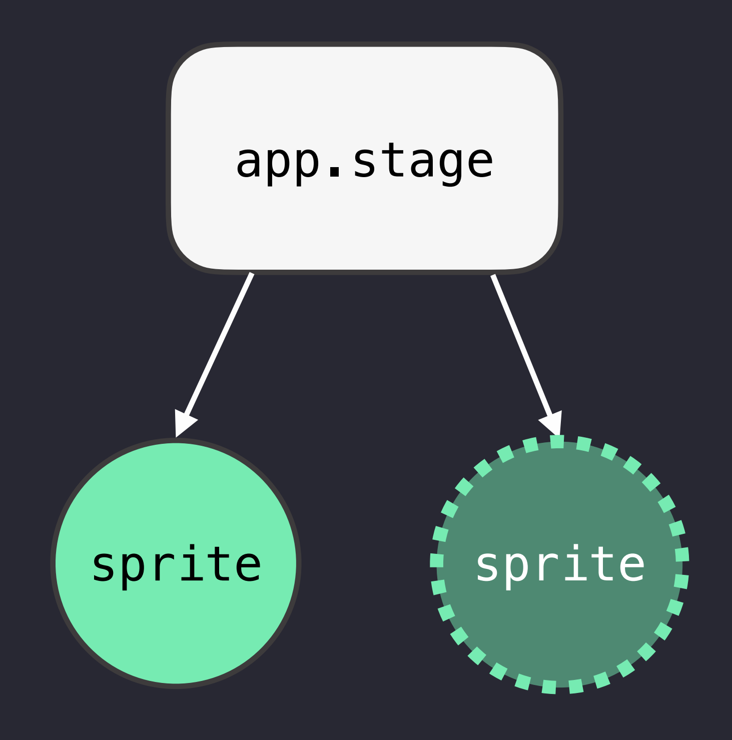







FC
Frontend
Conf 2022

Добавление в конец

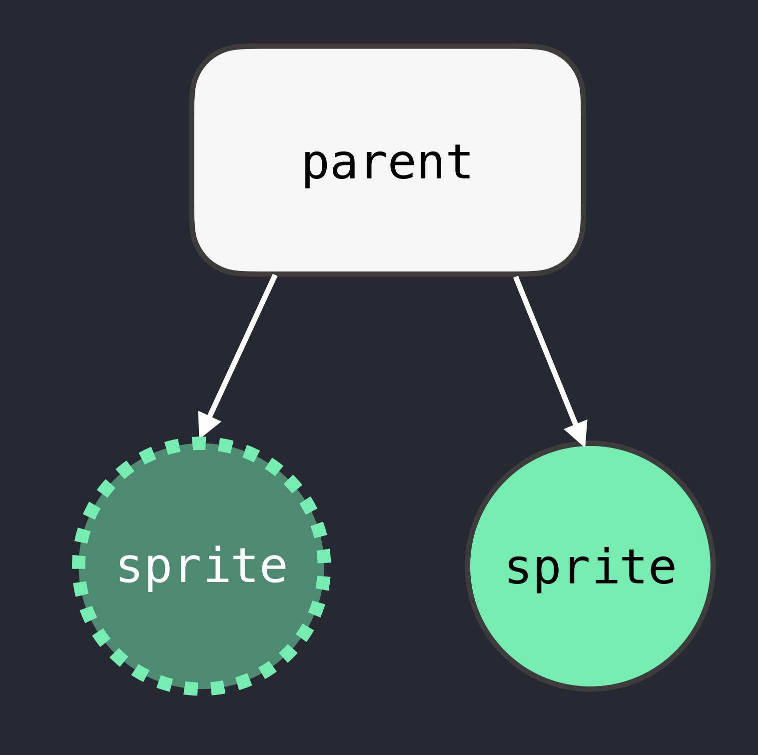



```
appendChild: (parent, child) => {
    parent.addChild(child);
  },
```

Добавляет ребёнка конец. Вызывается во время коммит-фазы



Добавление в начало




```
insertBefore: (parent, child, before) => {
    parent.addChild(child);
  },
```

Вставляет ребёнка перед некоторым узлом, который уже существует на экране, вызывается во время коммит-фазы



Добавление в начало

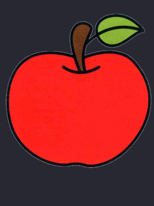
```
insertInContainerBefore: (container, child, before) => {
    container.addChild(child);
  },
```


Вставляет ребёнка перед некоторым узлом, который уже существует на экране, только для контейнера

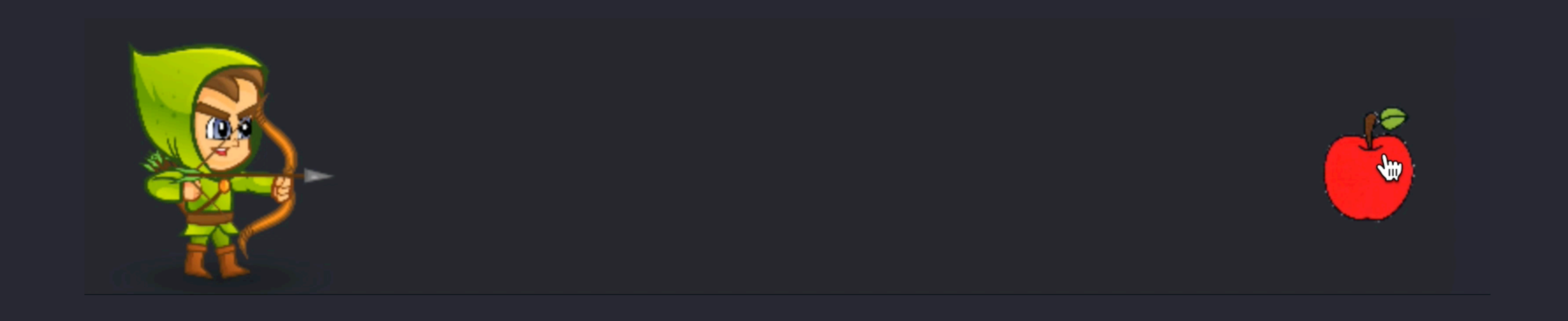


Демо










FC
Frontend
Conf 2022

Рефакторинг



Рефакторинг — Sprite

```
import * as PIXI from "pixi.js";

const Sprite = ({ img, ...args }) => {
  const texture = PIXI.Texture.from(img);

  return <sprite texture={texture} {...args} />;
};

export default Sprite;
```



Рефакторинг — Stage

```
import { useCallback } from "react";
import * as PIXI from "pixi.js";
import { render } from "../hostConfig";

const Stage = ({ children, width, height, options }) => {

  const mountStage = useCallback((canvas) => {
    const app = new PIXI.Application({ width, height, view: canvas, ...options });

    render(children, app.stage);

  }, []);

  return <canvas ref={mountStage} />;
};

export default Stage;
```



```
isPrimaryRenderer: false
```

```
If your renderer is used on top of React DOM or some other existing renderer, set it to false.
```



Рендер в ReactDOM

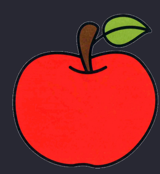
```
import Stage from "./components/Stage";
import Sprite from "./components/Sprite";
import img from "./resources/image.png";

import ReactDOM from "react-dom/client";

const container = document.getElementById("root");
const root = ReactDOM.createRoot(container);

root.render(<App />);

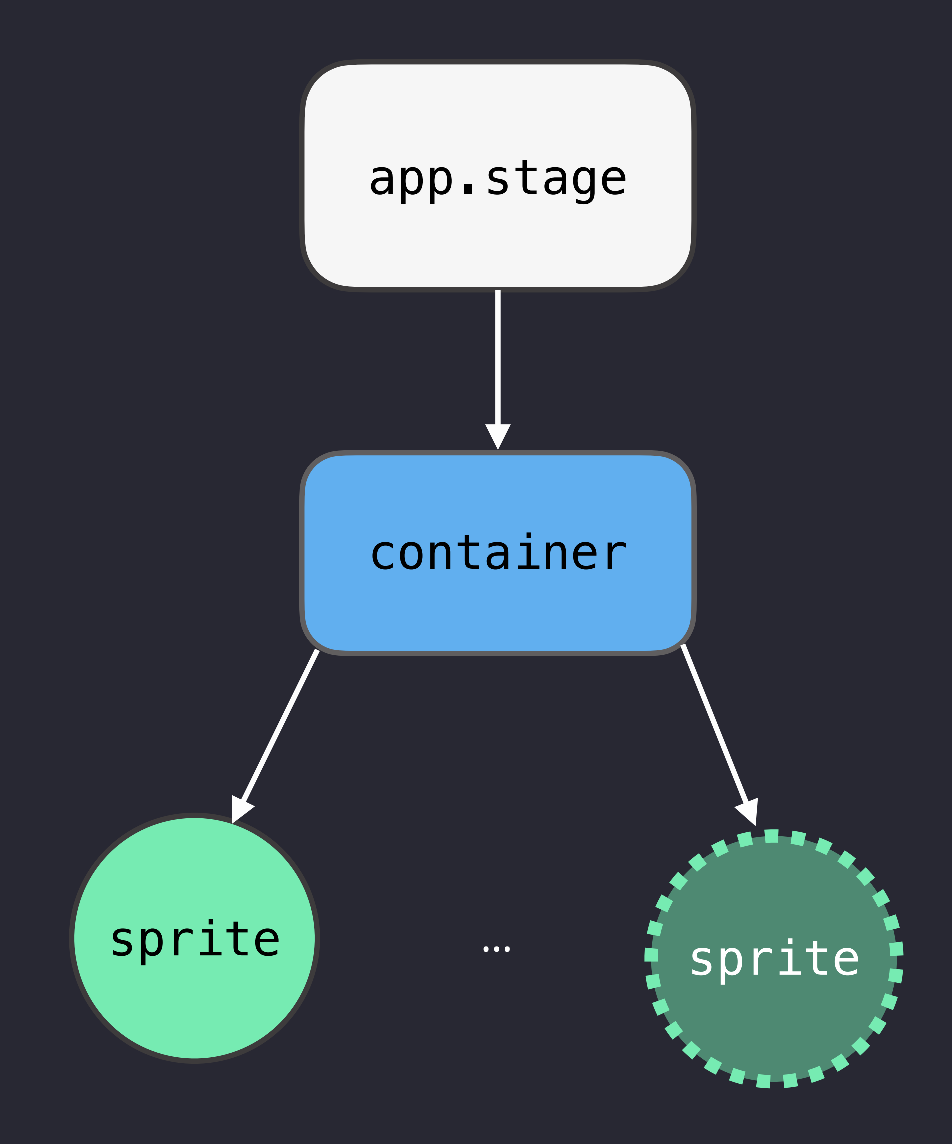
function App() {
  return (
    <Stage options={{ backgroundColor: 0x292c33 }}>
      <Sprite img={img} width={75} height={75} x={725} y={525} />
    </Stage>
  );
}
```





Анимация




app.stage
container
sprite
...
sprite


FC
Frontend
Conf 2022

```
1. getRootHostContext
```

app.stage

container

sprite

…

sprite



app.stage

container

sprite

...

sprite

1. getRootHostContext

2. **getChildHostContext**



app.stage

container

sprite

...

sprite

1. getRootHostContext

2. getChildHostContext

3. getChildHostContext



app.stage

container

sprite

...

sprite

1. getRootHostContext

2. getChildHostContext

3. getChildHostContext

4. **shouldSetTextContent**



app.stage

container

sprite

...

sprite

1. getRootHostContext

2. getChildHostContext

3. getChildHostContext
4. shouldSetTextContent
5. **prepareUpdate**



```
prepareUpdate: (instance, type, oldProps, newProps) => newProps
```

Проверяет есть ли изменения и находит их. Выполняется во время рендер фазы



app.stage

container

sprite … sprite

1. getRootHostContext
6. **prepareForCommit**

2. getChildHostContext

3. getChildHostContext
4. shouldSetTextContent
5. prepareUpdate



app.stage
container
sprite
...
sprite
1. getRootHostContext
6. prepareForCommit
2. getChildHostContext
3. getChildHostContext
4. shouldSetTextContent
5. prepareUpdate
7. commitUpdate


FC
Frontend
Conf 2022

```
commitUpdate: (instance, updatePayload, type, oldProps, newProps) => {},
```

Вносит изменения, найденные ранее. Вызывается в фазе коммита у всех элементов имеющих updatePayload



app.stage

container

sprite

...

sprite

1. getRootHostContext
6. prepareForCommit
8. **resetAfterCommit**

2. getChildHostContext

3. getChildHostContext
4. shouldSetTextContent
5. prepareUpdate
7. commitUpdate



app.stage

1. getRootHostContext
6. prepareForCommit
8. resetAfterCommit

container

2. getChildHostContext

sprite

…

sprite

3. getChildHostContext
4. shouldSetTextContent
5. prepareUpdate
7. commitUpdate



Пример

```
const container = document.getElementById("root");
const root = ReactDOM.createRoot(container);
root.render(
  <Stage options={{ backgroundColor: 0x292c33 }}>
    <App />
  </Stage>
);

function App() {
  const [x, setX] = useState(0);
  const handleMouseMove = (event) => {
    const x = Math.floor(event.data.global.x);
    setX(x);
  };

  return (
    <Sprite img={img} width={150} height={150} x={x} y={200} onMouseMove={handleMouseMove} />
  );
}
```



prepareUpdate

```
prepareUpdate: (instance, type, oldProps, newProps) => newProps,
```



commitUpdate

```
commitUpdate: (instance, updatePayload, type, oldProps, newProps) => {
  if (type === "sprite") {
    const { x = 0, y = 0, rotation } = updatePayload;

    instance.x = x;
    instance.y = y;

    if (rotation) {
      instance.rotation = rotation;
    }
  }
}
```






FC
Frontend
Conf 2022

Контекст и хук для анимации



Контекст и хук для анимации

```
app.ticker.add((delta) => {
    container.rotation -= 0.01 * delta;
});
```



Контекст и хук для анимации

```
import React from "react";

export const AppContext = React.createContext(null);

export const AppProvider = ({ app, children }) => (
  <AppContext.Provider value={app}>{children}</AppContext.Provider>
);
```



Контекст и хук для анимации

```
const Stage = ({ children, width, height, options }) => {
  const mountStage = useCallback((canvas) => {
    const app = new PIXI.Application({
      width,
      height,
      view: canvas,
      ...options,
    });

    const provider = <AppProvider app={app}>{children}</AppProvider>;
    render(provider, app.stage);
  }, []);

  return <canvas ref={mountStage} />;
};
```



Контекст и хук для анимации

```
export const useApp = () => {
    const app = useContext(AppContext);

    if (app === null) {
        return;
    }

    return app;
}
```



Контекст и хук для анимации

```
export const useTick = (fn) => {
    const { ticker } = useApp();

    useEffect(() => {
        ticker.add(fn);

        return () => {
            ticker.remove(fn)
        }
    }, [fn, ticker])
}
```



Пример с кодом мальчика

```
const container = document.getElementById("root");
const root = ReactDOM.createRoot(container);
root.render(
  <Stage options={{ backgroundColor: 0x292c33 }}>
    <App />
  </Stage>
);

function App() {
  const [x, setX] = useState(0);
  const [rotation, setRotation] = useState(0);
  useTick((delta) => {
    setRotation((rotation) => rotation - 0.03 * delta);
  });
  const handleMouseMove = (event) => {
    const x = Math.floor(event.data.global.x);
    setX(x);
  };

  return (
    <Sprite
      img={boy}
      width={150}
      height={150}
      x={x}
      y={250}
      rotation={rotation}
      onMouseMove={handleMouseMove}
    />
  );
}
```





Пример с кодом мальчика

```
const container = document.getElementById("root");
const root = ReactDOM.createRoot(container);
root.render(
  <Stage options={{ backgroundColor: 0x292c33 }}>
    <App />
  </Stage>
);

function App() {
  const [x, setX] = useState(0);
  const [rotation, setRotation] = useState(0);
  useTick((delta) => {
    setRotation((rotation) => rotation - 0.03 * delta);
  });
  const handleMouseMove = (event) => {
    const x = Math.floor(event.data.global.x);
    setX(x);
  };

  return (
    <Sprite
      img={boy}
      width={150}
      height={150}
      x={x}
      y={250}
      rotation={rotation}
      onMouseMove={handleMouseMove}
    />
  );
}
```





Пример с кодом мальчика

```
const container = document.getElementById("root");
const root = ReactDOM.createRoot(container);
root.render(
  <Stage options={{ backgroundColor: 0x292c33 }}>
    <App />
  </Stage>
);

function App() {
  const [x, setX] = useState(0);
  const [rotation, setRotation] = useState(0);
  useTick((delta) => {
    setRotation((rotation) => rotation - 0.03 * delta);
  });
  const handleMouseMove = (event) => {
    const x = Math.floor(event.data.global.x);
    setX(x);
  };

  return (
    <Sprite
      img={boy}
      width={150}
      height={150}
      x={x}
      y={250}
      rotation={rotation}
      onMouseMove={handleMouseMove}
    />
  );
}
```







Hunter





И всё же **смог**



Змейка



Builder



Builder

1. monaco-editor
2. <iframe />
3. Панель настроек

```
            setCounter((p) => p + 1);
            if (direction === 'up') setSnail((p) => [...p, { x: p[p.length - 1].
            if (direction === 'down') setSnail((p) => [...p, { x: p[p.length - 1
            if (direction === 'left') setSnail((p) => [...p, { x: p[p.length - 1
            if (direction === 'right') setSnail((p) => [...p, { x: p[p.length -
        }
    }, [direction, snail, screenWidth, screenHeight]);

    React.useEffect(() => {
        document.addEventListener('keydown', handleKeyDown);

        return () => {
            document.removeEventListener('keydown', handleKeyDown);
        };
    }, []);

    return (
        <>
            {background.map(({ x, y, number }, i) => {
                if (number === 1) {
                    return <ReactPIXI.Sprite key={i} img="../resources/trava.png
                } else if (number === 2) {
                    return <ReactPIXI.Sprite key={i} img="../resources/trava2.pn
                }
            })}
            <text text={counter} style={{ fontSize: 32, fill: 0xffffff }} x={38}

            {snail.map(({ x, y }, i) => {
                return <graphics key={i} fill={0x4287f5} drawRect={{ width: size
            })}
            <ReactPIXI.Sprite img="../resources/apple.png" width={50} height={35
            <ReactPIXI.Sprite img="../resources/apple.png" width={50} height={35
        </>
    );
};

const Component = () => {
    return (
        <ReactPIXI.Stage width={784} height={584} options={{ backgroundColor: 0x
            <PixiApp />
        </ReactPIXI.Stage>
    );
}
```

0

width:
50
height:
35

rotation:
0
img
../resources/apple.png
onClick
Добавить обработчик
children:
0
anchor:
0



Builder

1. monaco-editor
2. <iframe />
3. Панель настроек

```
        if (getIntersaction(position, applePosition, 30)) {
            const x = randomInteger(0, screenWidth - 50);
            const y = randomInteger(0, screenHeight - 50);

            setApplePosition({ x, y });
            setCounter((p) => p + 1);
            if (direction === 'up') setSnail((p) => [...p, { x: p[p.l
            if (direction === 'down') setSnail((p) => [...p, { x: p[p
            if (direction === 'left') setSnail((p) => [...p, { x: p[p
            if (direction === 'right') setSnail((p) => [...p, { x: p[
        }
    }, [direction, snail, screenWidth, screenHeight]);

    React.useEffect(() => {
        document.addEventListener('keydown', handleKeyDown);

        return () => {
            document.removeEventListener('keydown', handleKeyDown);
        };
    }, []);

    return (
        <>
            {background.map(({ x, y, number }, i) => {
                if (number === 1) {
                    return <ReactPIXI.Sprite key={i} img="../resource
                } else if (number === 2) {
                    return <ReactPIXI.Sprite key={i} img="../resource
                }
            })}
            <text text={counter} style={{ fontSize: 32, fill: 0xfffff

            {snail.map(({ x, y }, i) => {
                return <graphics key={i} fill={0x4287f5} drawRect={{
            })}
            <ReactPIXI.Sprite img="../resources/apple.png" width={50}
            <ReactPIXI.Sprite img="../resources/apple.png" width={50}
        </>
    );
};

const Component = () => {
    return (
        <ReactPIXI.Stage width={784} height={584} options={{ backgrou
            <PixiApp />
        </ReactPIXI.Stage>
    );
}
```




Recast



Recast

AST Explorer | Snippet | JavaScript | recast | Transform | default | ?

Parser: recast-0.21.1
Transformer: recast-0.21.1

```js
let tips = [1, 2, 3];

function printTips() {
  tips.forEach((tip, i) => console.log(`Tip ${i}:` + tip));
}
```

Tree | JSON | 12ms

☑ Autofocus ☑ Hide methods ☐ Hide empty keys ☐ Hide location data ☐ Hide type keys

```
- File {
  - program: Program {
      type: "Program"
    - body: [
      + VariableDeclaration {type, declarations, kind, range, loc}
      - FunctionDeclaration {
          type: "FunctionDeclaration"
        + id: Identifier {type, name, range, loc}
          params: [ ]
        - body: BlockStatement {
            type: "BlockStatement"
          - body: [
            + ExpressionStatement {type, expression, range, loc}
            ]
          + range: [2 elements]
          + loc: {start, end, lines, tokens, indent}
          }
          generator: false
          expression: false
          async: false
        + range: [2 elements]
        + loc: {start, end, lines, tokens, indent}
        }
      ]
      sourceType: "script"
    + range: [2 elements]
    + loc: {start, end, lines, indent, tokens}
    }
    name: null
  + loc: {start, end, lines, indent, tokens}
    type: "File"
```

FC Frontend Conf 2022

Recast

```
let tips = 1
```

```json
{
    "type": "VariableDeclaration",
    "declarations": [
        {
            "type": "VariableDeclarator",
            "id": {
                "type": "Identifier",
                "name": "tips",
                "range": [
                    4,
                    8
                ],
                "loc": {
                    "start": {
                        "line": 1,
                        "column": 4,
                        "token": 1
                    },
                    "end": {
                        "line": 1,
                        "column": 8,
                        "token": 2
                    },
                    "lines": {
                        "infos": [
                            {
                                "line": "let tips = 1",
                                "indent": 0,
                                "locked": false,
                                "sliceStart": 0,
                                "sliceEnd": 12
                            }
                        ],
                        "mappings": [],
                        "cachedSourceMap": null,
                        "length": 1,
                        "name": null
```



Recast

```
function foo(){}
```

```
{
    "type": "FunctionDeclaration",
    "id": {
      "type": "Identifier",
      "name": "foo",
      "range": [
        9,
        12
      ],
      "loc": {
        "start": {
          "line": 1,
          "column": 9,
          "token": 1
        },
        "end": {
          "line": 1,
          "column": 12,
          "token": 2
        }
      ...
    }
}
}
```



Recast

```
<App>Hello world</App>
```

```json
{
    "type": "ExpressionStatement",
    "expression": {
      "type": "JSXElement",
      "openingElement": {
        "type": "JSXOpeningElement",
        "name": {
          "type": "JSXIdentifier",
          "name": "App",
          "range": [
            1,
            4
          ],
          "loc": {
            "start": {
              "line": 1,
              "column": 1,
              "token": 1
            },
            "end": {
              "line": 1,
              "column": 4,
              "token": 2
            }
           ...
        }
    }
}
```



```jsx
<App>Hello world</App>
```

```ts
import { visit } from "recast";
import { parse } from "recast/parsers/babel";

export const getIndexByPos = (code: string, line: number, col: number): number => {
    try {
        const ast = parse(code);
        let index = 0;
        visit(ast, {
            visitJSXOpeningElement(element) {
                if (
                    line >= Number(element.value.loc.start.line) &&
                   col >= Number(element.value.loc.start.column) &&
                    line <= Number(element.value.loc.end.line) &&
                   col <= Number(element.value.loc.end.column)) {
                    this.abort();
                }
                index++;
                return false;
            }
        });

        return index;
    }
    catch (error) {
        return -1;
    }
};
```


```json
{
    "type": "ExpressionStatement",
    "expression": {
      "type": "JSXElement",
      "openingElement": {
        "type": "JSXOpeningElement",
        "name": {
          "type": "JSXIdentifier",
          "name": "App",
          "range": [
            1,
            4
          ],
          "loc": {
            "start": {
              "line": 1,
              "column": 1,
              "token": 1
            },
            "end": {
              "line": 1,
              "column": 4,
              "token": 2
            }
          ...
        }
    }
}
```


FC Frontend Conf 2022

```
<App>Hello world</App>
```

```
import { visit } from "recast";
import { parse } from "recast/parsers/babel";

export const getIndexByPos = (code: string, line: number, col: number): number => {
    try {
        const ast = parse(code);
        let index = 0;
        visit(ast, {
            visitJSXOpeningElement(element) {
                if (
                    line >= Number(element.value.loc.start.line) &&
                    col >= Number(element.value.loc.start.column) &&
                    line <= Number(element.value.loc.end.line) &&
                    col <= Number(element.value.loc.end.column)) {
                    this.abort();
                }
                index++;
                return false;
            }
        });

        return index;
    }
    catch (error) {
        return -1;
    }
};
```



```
<App>Hello world</App>
```

```
import { visit } from "recast";
import { parse } from "recast/parsers/babel";

export const getIndexByPos = (code: string, line: number, col: number): number => {
    try {
        const ast = parse(code);
        let index = 0;
        visit(ast, {
            visitJSXOpeningElement(element) {
                if (
                    line >= Number(element.value.loc.start.line) &&
                   col >= Number(element.value.loc.start.column) &&
                    line <= Number(element.value.loc.end.line) &&
                   col <= Number(element.value.loc.end.column)) {
                    this.abort();
                }
                index++;
                return false;
            }
        });

        return index;
    }
    catch (error) {
        return -1;
    }
};
```



```
import { print } from "recast";
import { transform } from "@babel/standalone";

const code = print(ast).code;

const transformed = transform(code, {
    presets: ["env", "react", "tsx"],
});
```



```
import { print } from "recast";
import { transform } from "@babel/standalone";

const code = print(ast).code;

const transformed = transform(code, {
    presets: ["env", "react", "tsx"],
});
```





Демо





Интересные проекты



React-pixi-render

2D-графика

https://github.com/deeamtee/react-pixi-render









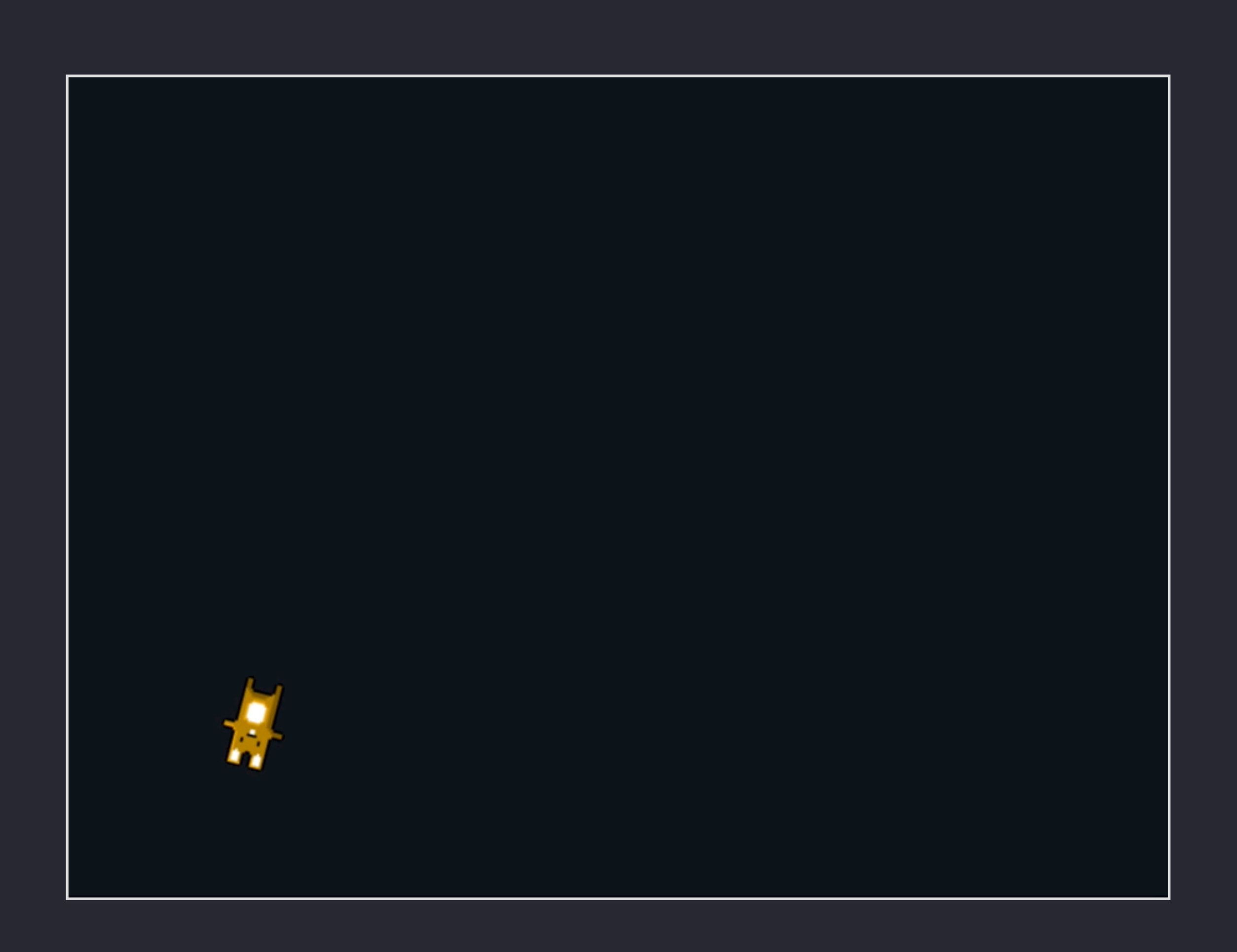
React-pixi

2D-графика

https://github.com/inlet/react-pixi

https://reactpixi.org/







React-pixi-fiber

2D-графика

https://github.com/michalochman/react-pixi-fiber

https://pylnata.github.io/mowgli/

https://github.com/Tinkoff/react-pixi-racing-game



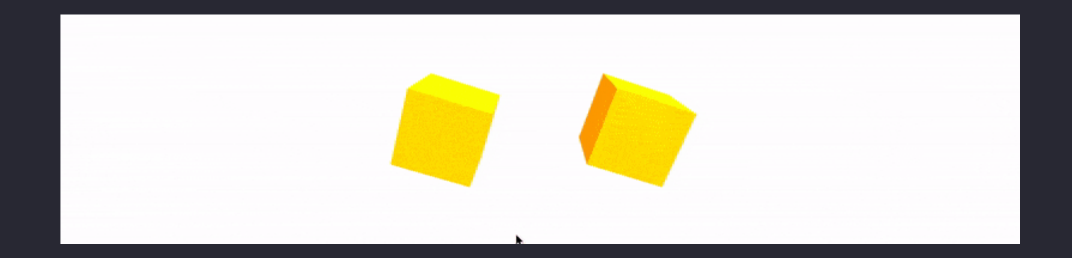




React-three-fiber

3D-графика

https://github.com/pmndrs/react-three-fiber



```
import { createRoot } from 'react-dom/client'
import React, { useRef, useState } from 'react'
import { Canvas, useFrame } from '@react-three/fiber'

function Box(props) {
  // This reference will give us direct access to the mesh
  const mesh = useRef()
  // Set up state for the hovered and active state
  const [hovered, setHover] = useState(false)
  const [active, setActive] = useState(false)
  // Subscribe this component to the render-loop, rotate the mesh every frame
  useFrame((state, delta) => (mesh.current.rotation.x += 0.01))
  // Return view, these are regular three.js elements expressed in JSX
  return (
    <mesh
      {...props}
      ref={mesh}
      scale={active ? 1.5 : 1}
      onClick={(event) => setActive(!active)}
      onPointerOver={(event) => setHover(true)}
      onPointerOut={(event) => setHover(false)}>
      <boxGeometry args={[1, 1, 1]} />
      <meshStandardMaterial color={hovered ? 'hotpink' : 'orange'} />
    </mesh>
  )
}

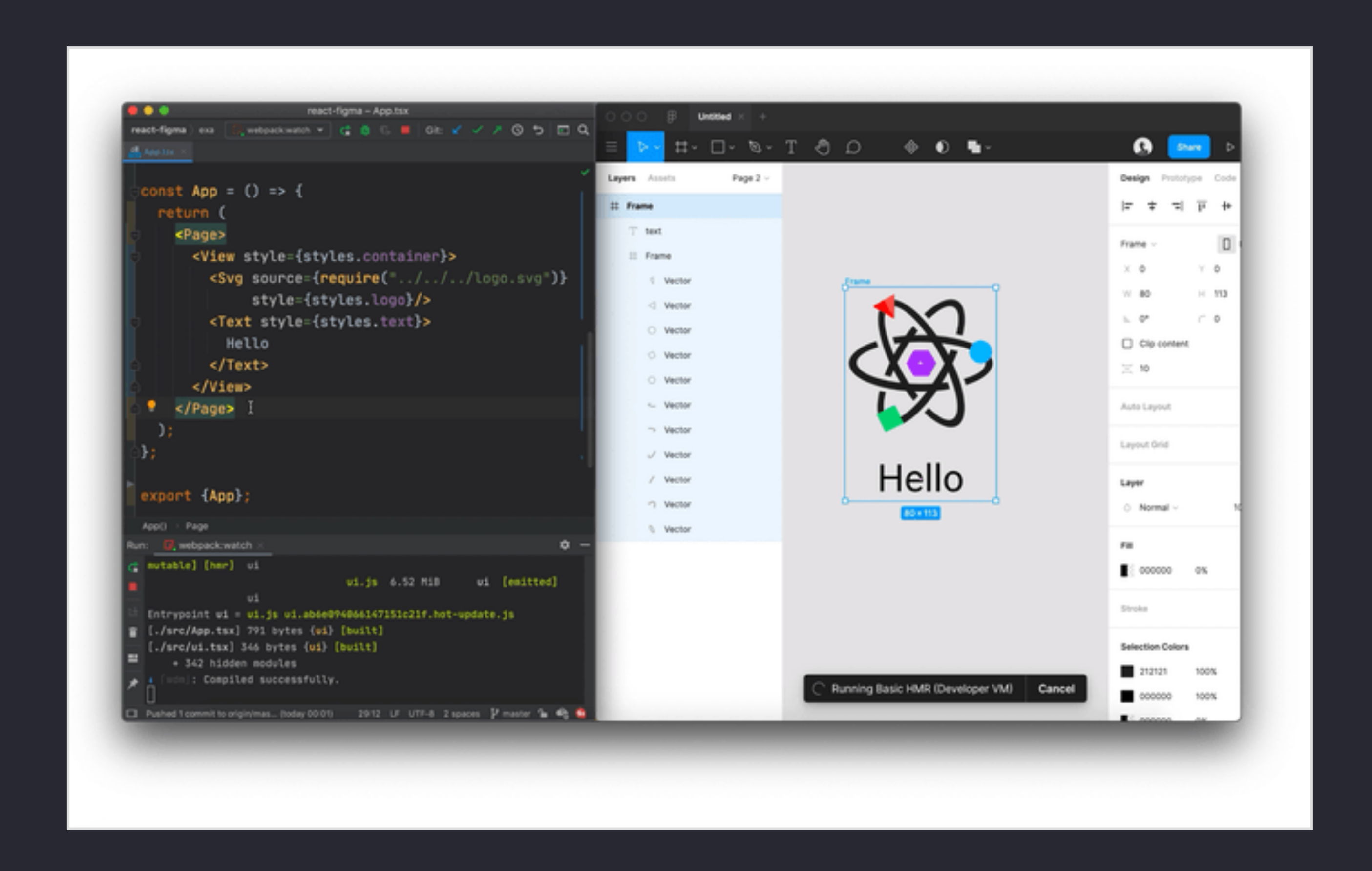
createRoot(document.getElementById('root')).render(
  <Canvas>
    <ambientLight />
    <pointLight position={[10, 10, 10]} />
    <Box position={[-1.2, 0, 0]} />
    <Box position={[1.2, 0, 0]} />
  </Canvas>,
)
```



React-figma

Design

https://github.com/react-figma/react-figma



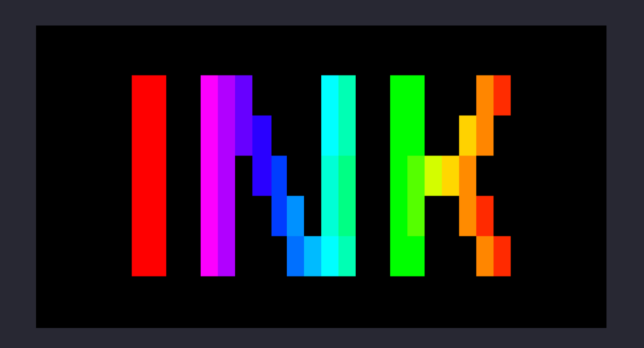


Ink

Command line Interfaces

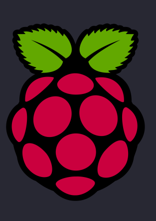
https://github.com/vadimdemedes/ink

```
~/Projects/ink
❯ node media/example
0 tests passed
```





React-hardware



Hardware

https://github.com/iamdustan/react-hardware





Awesome-react-renderer

Web (+ NW & Electron)
3D
Desktop
Mobile
Command Line Interface
Television
Hardware
Email
File
Design
Music
Chatbot
Miscellaneous
etc.

https://github.com/chentsulin/awesome-react-renderer





Tetragius/novis

Game

Игра на промисах со сценариями в yaml

https://github.com/Tetragius/novis





React-world

Game

Игра на ReactDOM. Анимации на CSS

https://github.com/sfatihk/react-world

https://sfatihk.github.io/react-world/





Итоги

1. Познакомились с работой react-reconciler
2. Написали простой движок, рисующий с PIXI.js
3. Собрали систему для создания игр кликами
4. Интересные проекты



Да прибудет с вами React!



QR-код с формой обратной связи





